sql Finance

User Manual

sqlFinance - Enterprise Resource Planning (ERP)

sqlFinance

User Manual

I. INTRODUCTION 3

1. SHORT OVERVIEW 3
2. ACCOUNTING SYSTEM 5
   - A. DOUBLE ENTRY BOOK-KEEPING 5
   - B. THE CONTRIBUTION OF LUCA PACIOLI (1494) 5
   - C. NORMALISATION OF DATA 7
   - D. THE DATA MODEL OF SQLFINANCE 10
   - E. PROCESS ORIENTED ACCOUNTING 11

II. QUICK START 13

1. INSTALLATION GUIDE 13
2. START WITH A NEW MANDATE 15
3. APPLICATION OVERVIEW 18
4. SINGLE AND DOUBLE SIDED BOOKINGS 27
5. DATA MANAGEMENT 31

III. APPLICATION MODULES 47

1. GENERAL LEDGER 47
2. A SIMPLE SALES TRANSACTION 49
3. DEBIT/CREDIT CLEARING 53

IV. KEY FUNCTIONS 63

1. FORMS AND CASCADES 63
2. FIELDS 66
3. MENUS 68
4. DRAG AND DROP 72
5. REPORTS 76
6. KEYBOARD SHORTCUTS 78

V. SUPPORT & CONTACT 79

VI. APPENDIX 80

sqlFinance

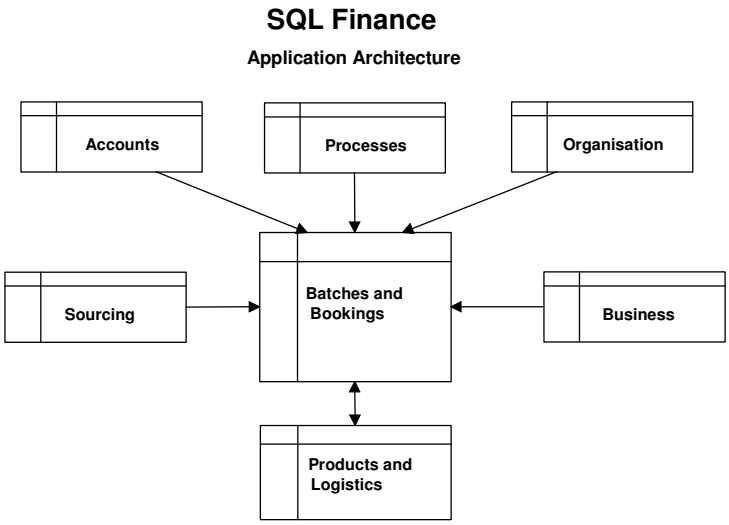
User Manual

I. Introduction

1. Short Overview

In **human resources consulting**, the management of **customer** and **candidate contacts** is a key factor. This is true for the **quality** of the interaction between the **consultant** and his **contact partner**, but also for the **logistics** used in the **administration process**. sqlFinance originally was designed in this context and **migrated** in several steps from the first **UNIX** based database over to a **process** oriented **client server application**.

Starting from basic features like **Customer Relationship management** and **batch processing**, sqlFinance has been further enhanced to a comprehensive software system for the management of **business data**. The **object-oriented interface** allows the **simultaneous access** of separate **local and server databases**. Several **mandates** can be defined in one database, with different accounting charts but shared **customer data**. Decentralised server installations may exchange their data through **replication tools**.



Through its **innovative approach** sqlFinance organises the general ledger accounting according to **processes**. Similar **bookings**, grouped into **batches**, get related to **processes**. This leads to considerable **rationalisation effects**, not only at the moment of **data entry**, but also on the **reporting** and **auditing** level.

For the **data entry**, similar **booking patterns** can be easily copied and reused. With use of **drag and drop** new batches and bookings can be **generated** from older ones and the **adjusted** according to user needs. **Reports** based on these

processes give an **efficient overview** of the entered **booking patterns** and can be used to verify the **consistency** of the data and identify any errors.

Similarly, the **auditor** will be able to check the accounting **process by process**. Each process is a **collection** of **similar batches** and must result in a consistent **balance** and **profit/loss statement**. In the **"process"** data structure, it is also possible to register **accounting rules** and **directions** for the bookkeeping. The **auditor** then will be able to check process by process the **bookings** against the **established rules**.

Furthermore**, sqlFinance** structures booking data according to **contract partners**, as well as **business fields** and **organisational units** ("contract", "business", "organisation"). All possible **business relations** can be managed as **"contracts"**: e.g. clients, suppliers, employees, but also bank accounts, rental agreements, insurance policies, etc. The **"business"** field rather represents the **business activity**, e.g. major projects, market segments for distribution, supply areas. While the **"business"** is related to the **external activit**y, the **organisational unit** is related to the **internal structures**, like profit or cost centers.

Based on the **general ledger data**, an efficient **controlling** can be created. According to your needs you can emphasize your analysis more on the control of the **contract relationships**, the **performance of business fields** or the **productivity of the organisational units**.

sqlFinance includes reports for open and already closed periods, according to the needs of the user. At the level of booking entry **foreign currencies**, **rounding**, **charges** and **commissions**, but also **TVA tax** can be entered. A module for the **payroll** support **recurring payments** to employees, as well as automatic **generation of bookings** for the calculation of **social benefit expenses**. Finally, the **time management** module will allow you to allocate **labour expenditure** to business fields and organisational units.

In the **sales** and **purchase** modules, **products** can be managed, with **quantities**, and integrated into **bills of material**. The **billing** of the sales module allows you to use different **tax rates** in the same bill. Different **addresses** can be used for **billing** and **shipping**. In the **server version**, **inventory position**s and the tracking of **stock movements** is also possible.

Language management has been considered a key feature from the beginning. All language is driven from **databases**, and a Language Manager **administration tool** allows the efficient **adaptation** to another language.

An **easy start** is offered with the **free download** of the **general ledger** module. This version includes an **IFRS** chart of accounts. **Foreign currency** support is included, as well as **organisation** (cost centre) and **contract partner** definition at booking level. The **free download** version is limited to **one simultaneous user**.

sqlFinance has been used for companies audited by

PRICEWATERHOUSECOOPERS

in **Switzerland**. Revisions also have been effected by Social Security and official Tax Auditors. The software has met – without exception – **all requirements** of these audits. In a recent review by **Prof. Dr. Beatrice Meyer** of the **Winterthur School of Management** the application and software design found **unconditional approval** and were considered as an **innovative** and **interesting approach**.

2. Accounting System

a. Double entry book-keeping

Accounting is **mathematics**. Most **simple** mathematics one could think. The most frequent operation is the **addition**: revenues and expenses are added separately, the **difference** results in the **profit**. Just because this seems so easy, there is little indication in the technical literature, what really are the **mathematical requirements** of an **accounting system**.

Furthermore, there are different **booking styles** found in practice, that differ mainly in the way data is **recorded**. In **Europe**, it is common to register on the **booking line** not only an **account** and an **amount**, but also a **contra-account**. In the **United States** however, the **simple booking line** with just an **account** and an **amount** is widespread used. In some cases, the booking lines are associated to a **common contra-account** (e.g. receivables clearing account), what gives a **mixed style** positioned between the two main data entry modes.

We would like to call the first variant **„double sided booking"** or „DSB", the second variant **„single sided booking"** or „SSB"). **sqlFinance** accepts **both ways** of data entry. It is even possible, to use both booking styles within the **same mandate**. Thus, in a **first period** you can continue with the **"double sided booking"** style, and then **step by step** get acquainted with the advantages of the **"single sided booking"** style.

"Double entry bookkeeping" or **„double entry accounting"**) has its **roots** in the **Middle Ages**. At this time, all data was registered **manually** on **paper**. It is evident that the availability of **information technology** and especially the use of **relational databases** has opened **other possibilities** for an **efficient accounting system**.

In order to meet the **requirements** of the **regulators** and the **clients** often insecure in this regard many software suppliers have integrated the **old habits** in their applications. However, if you want to use more the **potential** of today's **information technology** for your accounting, it is necessary to consider the **mathematical requirements** of an accounting system and **review** the **traditional way** of bookkeeping. A **serious evaluation** shows that there are **several solutions** available, each one with **advantages** and **disadvantages** and their practical implications.

Thus, it seems worth wile to consider the **historical development** of **„double entry bookkeeping"** and the evolution of **mathematical requirements** for **accounting systems**.

b. The contribution of Luca Pacioli (1494)

The historian agree that **double sided booking** has been initiated by Franciscan father **Luca Pacioli** (1445-1517) and his publication from the year **1494**. In his act **„Summa"** the mathematician has written a summary of the **mathematical methods** of **his times**. In the famous Tractatus XI with the title **„Particularis de computis et scripturis"** Pacioli describes the **rules of bookkeeping** as they were applied at that time by the northern Italian **trading houses**.

Typical for the bookkeeping of these times is a **high redundancy** in the **data collection**. The business event was **first registered** as a note in a **Memorial**. Based on this protocol a **financial expert** established a **booking** with **principal and believer** (debtor and creditor) and entered it in the journal. In a third step this **booking** was **transferred** to the general ledger, which included a separate account sheet for each **principal** und **believer**.

Luca Pacioli: Summa de arithmetica, geometria, proportioni et proportionalità. Toscolano, 1523 (2. Ed., Signature: 72520)

Separate **account sheets** where used not only for the **settlement** with the **contract partners** (clients and suppliers), but also for the **inventory control** of the **trading goods**. Furthermore **operating expenses** were summarized on some accounts, as well as **cash positions** on the **principal** and the engaged **capital** on the **believer** side.

Through the use of the **booking line** as an **intermediate element** the accountant attempted to assure that every change on the **principal side** had its **counterpart** on the **believer side**. If any **deviations** were found, they had to be reported as **profit or loss**, get attributed to the **business event** and moved to a summary account "profits and losses". At the moment of **closing**, this account was **cleared** with the owner's equity. Such deviations **arose** typically **from the inventory**, when the **sales price** was **higher** than the **purchase price,** and the **difference** immediately to be attributed to the **profit**.

Therefore, the **goal** of the bookkeeping was not just to **keep principals and believers in a balanced position**, but rather to **calculate the profit for each business case** and attribute it immediately to the **profit and loss account** or the **owner's equity account** respectively. It is interesting to note that **Pacioli** did not present any propositions for the **summarisation** of the profits or losses **per client**, what would have given the base of a **simple controlling**. It is possible that this kind of **view** on the **client relation** at that time was **not considered desirable** yet.

According to **Pacioli** the accounting was **„in balance"**, if the **sum** of the **principal** as well as the **believer side** separately added would give the **same (positive) amount**. Were the amounts of the **believer side** considered as **negative values**, then the accounting was **"in balance"**, if the **sum of all principal and believer values** resulted in a **total of zero**.

Mathematically, the **double sided accounting** of the **Middle Ages** can be considered as a list of **booking values**, whose **total** resulted in an **amount of**

zero. Are the **positive amounts** attributed to the collective name **"assets"** and the **negative amounts** to the collective name **"liabilities"** we get a **balance** with a **total of zero**.

In **Pacioli**'s work already is apparent, how **profits and losses** emerge from **not furthermore explainable changes** in value. As **Paul Weilenmann** (Grundlagen des betriebswirtschaftlichen Rechnungswesens, 1988) brings it to the point, it is seems reasonable to **separate not furthermore explainable asset gains and losses from the balance** and attribute them as **expenses and revenues** to a **separate income statement**. This way we get to the typical **partitioning** of today with **assets**, **liabilities**, **expenses** and **earnings**.

As soon as we allow bookings between **balance** and **income statement** emerges a **difference** between **assets and liabilities** and the **inverted difference** between **expenses and earnings**. This difference we denominate as **profit or loss respectively**. **It is the sum of the not furthermore explainnable difference between expenses and earnings** (see **Paul Weilenmann**, ibidem).

Balance	**Income statement**
explainable gains and losses	not furthermore explainable gains and losses
(1) + assets	(3) + expenses
(2) - liabilities	(4) - earnings
= **+ profit**(balance)	**= - profit** (income statement)

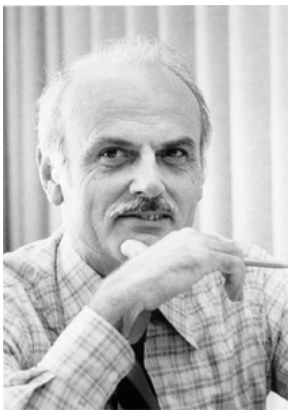
Balance and income statement with **explainable** and **not furthermore explainable** gains and losses

The **accounting system** is **defined** as a **list** of **at least an account and an amount**, with a **total of zero** for **all amounts** of the list, whereas the **accounts** are attributed to the **four types** (assets, liabilities, expenses and earnings), and the **profit** is calculated in the **balance** as the sum of **assets + liabilities**, and the **negative profit amount** in the **income statement** as the **sum** of **expenses + earnings**. **Bookings** can bee **added in any way**, under the **sole condition** that the **sum** of the new bookings would also result in a **total of zero**.

c. Normalisation of data

An **accounting system** is as good as the underlying **data model**. A consistent **data organisation** simplifies not only the **data entry**, but enhances the flexibility and significance of the **reports** as well. It is surprising, how little **data modelling** has been recognized as a possibility for **innovations** in the field of **general ledger accounting**.

The **organisation** of data according to **mathematical criteria** is a **prerequisite** for the efficient use of databases and is know under the term of **"normalisation"**. The **basic concepts** have been introduced by **Edgar F. Codd** (E.F.Codd, „A Relational Model of Data for Large Shared Data Banks", Comm.ACM 13 (6), June 1970, pp. 377-387). Working with large databases **Codd** developed his **rules of normalisation** that are widely accepted up to this day.



Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* 13 (6): 377–387

Codd discovered soon that through **normalisation** of data a **universal data language** could be developed. With this approach the **application** was able to reach a **maximum of independency** from the **data**. Furthermore, to a large extent the **redundancy** could be reduced and **multiple entries** requiring **complicated administration** tools again **avoided**.

In **practice** the **maximum elimination of redundancy** is **not consequently realised** however. An important **reason** is the **time aspect**. In a **billing process** for example, it is necessary to register the client address **each time** a document is sent. Considering data consistency it would **not be desirable** that a **change of address** would apply to **all open or even posted** billings.

Another **limiting factor** is the **complexity of the application**. Consequently **normalised data** leads to **many very small tables** that are **reintegrated** for the application into so called **"views"**. The usage of **views** requires a **technical basis** for this in the used **database**, an element **rather complicated** to realise.

As a **third element** should be considered that **Codd**'s reflections treat mainly the structuring of **master data**, basically composed of an **index** and **descriptive elements** (e.g. address number and address data). The **key question** for **double sided accounting** regarding **multiple indices in the same transaction table** remains here **not answered**.

Following the arguments of **Codd**, the further **normalisation** of **double sided** into **single sided bookings** seems an **evident conclusion**. At the **first normalisation level** already **Codd** required that the **repetition of homogenous data** is to be **avoided**:

Instead of a **list** with **structure A**:

- batch, account1, account2, amount

It is better to establish a **list** with **structure B:**

- batch, account, amount

For the **accounting** this is especially **evident**, since the **bookings** are **attributed to a batch**. Following the list with **structure A** there are immediately **problems arising** for the **reporting** programs. With such a **data structure**, how could a database selection call for a **balance** or an **income statement** ever work **efficiently**? For **every query line** the program would have to **check** if account1 **or** account2 would **correspond** to the **current reporting account**, and for **account1** the **positive amount** and for **account2** the **negative amount** would have to be considered for the **calculation** of the **report total**.

In a more general view, this **design issue** is well known among programmers as the so called **"bill of materials"** problem. Records of a **bill of materials** include **two product numbers**, one for the **master product** and one for the **detail product**. Two product numbers are necessary, since **each product** can be relied in **multiple ways** to **any other product** (compare to this subject: **Carl August Zehnder**, **Informationssysteme und Datenbanken**, B.G.Teubner, Stuttgart 1989, mainly **pages 60 f**).

For the full **normalisation** of this **„many to many" relation** the **relationship** as a **container** must be **registered** in the database as well. Based on this entry, **single sided relations** to the **products** can be entered. In **practice** this rather complicated solution is **often avoided**, since it is **not efficient** for applications with **drag and drop** functionality, where for example a **master product** is simply **dragged** over a **detail product**. The application can be **simplified** recording the **master product** as **product1**, the **detail product** as **product2**. With this approach the **reports** however get **more complicated**. These were probably the considerations that got **Pacioli** and others to use the **double sided bookings**.

On the other side, with today's **technical possibilities**, there is **little interest** to **omit the "batch"** in the **accounting data model**. On the **contrary**, for the **revision** and the **comprehension** of the **booking process** the possibility to **attribute bookings to a batch** is **most valuable**. This **corresponds well** to the **usual presentation on paper**. Thereby it is **less important** if the bookings are **single or double sided**. It is **only necessary** to assure that the **sum of all bookings** in the batch result in **a total of zero**. For **double sided bookings** this will **always** be the case, for **single sided bookings** it will be necessary to **check** for the zero total **before** the batch can be posted.

It is also interesting to reflect the **account assignment** under the **aspect of normalisation**. As a matter of fact, almost all **charts of accounts** include **several dimensions** summarized to an **overloaded register** difficult to handle. A **chart of accounts** often **includes elements** to structure the **internal organisation**, the **business field** and the **contract partner**. This makes the **general ledger** too complex and requires considerable efforts to **separate** the chart of accounts into these **underlying dimensions** in the reports.

The **complexity** of the **chart of accounts** is already apparent in the proceedings of **Pacioli**. The **initial concept** of an **account** was to represent a

client reporting (that is the **contractual relation** to the **principal** or the **believer**). But soon enough **virtual principals and believers** appear, like **merchandise purchase and stock**, and later **general expenditures** as well as **profits and losses** make the **contractual aspect** of the account **completely disappear**.

These arguments show that it is much **more efficient** to use a **consistent data model** for the **general ledger accounting as well**. This way, the **chart of accounts** can be kept **slim** and **oriented** for the **financial reporting**. The details for the **contract partners**, **business fields** and **organisational units** are then available from **supplementary data structures** at the user's demand.

d. The data model of sqlFinance

The **data model** of **sqlFinance** provides an excellent **base** for later analysis and an efficient c**ontrolling**. Normalised bookings are very suitable for the treatment with statistical methods and can be considered as a **multivariate system.** A number of **parameters** (account, organisation, contract, etc.) influences as a **resulting variable** the **amount**. This way we get straightaway a correctly designed **data-Cube**, as it is common in **„business intelligence"** applications. This data cube will include the following **parameters**:

- **Organisation**: the **internal** organisation of the company, often addressed as **cost centre** structure
- **Business**: the single **client project** or a **collection** of **client oriented activities** (in this case: **business field**)
- **Contract**: the **contract partner** (client, supplier, employee)
- **Product**: the **physical product** or the **service**, with reference to **product lines**, **measurement units** and **prices**

The possibility to indicate a **contract partner** in all bookings makes it possible to keep the **chart of accounts** very **slim**. Since the **contract partners** are transferred to a **separate reference table**, the **chart of accounts** only holds the stable **scaffold** for the **presentation** in the **balance** and the **income statement**. The **contract number** can be **equally used** for the administration of **bank accounts**, **insurance policies**, **telephone numbers**, **rental agreements**, etc. In addition to this **contract reference**, there is the **supplementary possibility** to manage **client projects** or **internal projects** in a further reference table (**"business"**). With this approach **single sided bookings** can attribute **supplier movements** directly to **client projects**, which makes the **internal logistics** and **project reporting** very **efficient**.

While the **contract partners** represent the **external relations** of the company, the **organisation** serves for the structuring of the **internal relations** and responsibilities. The **organisation** can represent a **cost** or a **profit centre**. **Organisational units** can be assigned to **master organisations** (e.g. departments or sales areas). Similarly **employees** can be registered as **part** of an **organisational unit**.

The **product field** is only available in the modules **purchase** and **sales** and offers not only the possibility to **manage material and services**, but also entire **bills of materials** and **inventory locations**.

In the **full version sqlFinance** includes a **payroll** and **time management module** as well. The **salaries** can be attributed to **organisational units**. The **partial closing** of an **intermediate period** will then show the **personnel expenses** per **organisational units** and import these **summary bookings** into the **general ledger**. The same way **operating hours per employee** can be attributed to a **project** or **business field**, or get moved from **one organisational unit** to **another one**. Bookings of this kind can be **imported** as a **summary** to the **general ledger reporting** as well.

e. Process oriented accounting

If you compare **registered batches** of an **accounting system**, you will soon recognize that the **booking patterns** are frequently repeated. Receivables, payables, payments, payroll, bank interests and expenses, depreciation, tax: all these **processes** follow a particular **booking pattern**. For the **accountant** as well as the **auditor** it seems rewarding to analyse and verify not every singular batch, but **batches grouped into processes**.

Felder | Liste

process	table_name	type	header
audit	PROCESS	P	external audit
booking	PROCESS	P	master booking pr
business	PROCESS	P	master business p
cash	PROCESS	P	cash account in ba
clear	PROCESS	P	clearing
closing	PROCESS	P	closing in/to gen.le
consulting	PROCESS	P	consulting services
credit	PROCESS	P	accounts payable
customer	PROCESS	P	customers (sales)
debit	PROCESS	P	accounts receivabl
default	PROCESS	P	default process
document	PROCESS	P	documents

Example of a process structure in a process window of sqlFinance

For the **assignment of bookings** to **batches and processes single sided** as well as **double sided bookings** can be used. It is only necessary to make sure that the **sum of all entered bookings** result in a **total of zero**. With **double sided bookings** this is **always** the case, since every booking line will add the **positive amount** for the **debit** and the same but **negative** amount for the **credit account**. With **single sided bookings**, the user can **only commit** the batch if the **zero total condition** is **fulfilled**.

With today's **information technology resources** it is **not really difficult** to **check batches** for a **total of zero** over all bookings. **Incorrect entries** can be **easily identified** and modified. The **batch** can also be used for the **preliminary register** of still **incomplete data** and be committed at a **later moment** of time to the accounting system.

Typical **processes** to group **batches** are e.g. purchase and sales, payments, interest calculations, capital increase, but also **processes not relied to a cash movement** like depreciation, establishment of reserves, etc. The **accountant** as well as the **auditor** will **gain on efficiency** if they analyse and verify **not only every batch**, but **entire processes**. For each **process** can be established a

small balance and **income statement**. Since the **process** follows a **homogenous pattern**, the **report** will only include **very few accounts**. For this reason **incorrect bookings** are **easily identified**. For **each process**, a **profit or loss** can be calculated as well, and the **addition** of the total **company profit** be tracked over **all processes** involved. In this context it may be valuable to differ between **two process classes** grouping **cash relevant batches** (e.g. interest, expenses) on one side and **batches with calculatory bookings** (e.g. depreciation) on the other.

Certain processes (like the payroll) can be set up **outside the general ledger**. This allows it to **reduce considerably** the **data volume** of the **general ledger**, and the **functionality** of the **outside application** can be **optimised** in **complete independence** of the **requirements of the general ledger**. In all these cases, the **process oriented data structure** of **sqlFinance** offers an **optimal interface** to **import any data**, with **single sided bookings** collected in **batches**. The **only condition** is here again that the **total of all bookings result in zero**.

Mainly at the moment of **revision** the efficiency of **process oriented financial accounting** becomes **evident**. The **auditor** does not need any more to **reconstruct** the **business event** out of the **journal**. In the opposite, he can **audit groups of batches summarised in processes**. For the auditor more **interesting processes** (like depreciation and other adjustments) are easily **identified**. In the **notes** of the batches **comments of the accountant** may be registered. These **notes and the associated proceedings** can then be **compared** with **instructions registered on the process level**, in order to **verify the consistency** of the financial reporting according to the given **rules and regulations**.

II. Quick Start

1. Installation Guide

Instructions for the installation

The software **sqlFinance** can be downloaded from the homepage (http://www.sqlfinance.com). In the **download center** not only the software, but also instructions and a manual are available for the user.

You choose between the **sqlFinance "public version"** (free version) or the **sqlFinance "full version"**. As soon as you have selected a version, you can **fill out** an **order form**. You enter your personal data and then receive **by email** the **link** for downloading. In accordance with the instructions you can **install** now **sqlFinance**. The default path ("**c:\programs\sqlFinance**") is defined in your operating system and can be altered during the installation process.

Notes for the installation of the full version of sqlFinance

There are important differences between the sqlFinance **"public version"** and the sqlFinance **"full version"**. A short list with the **features** of the two versions follows below:

The **free version** („public version") is limited to **single user mode** and offers:

- financial accounting with unrestricted number of mandates and subperiods

The **full version** of supports **multi user mode** with local and network **database** access (including **Microsoft sqlServer**) and includes the following **application modules**:

- purchase and sales orders
- inventory control and shipping
- accounts receivable and payable
- human resources and payroll
- time management
- portfolio management

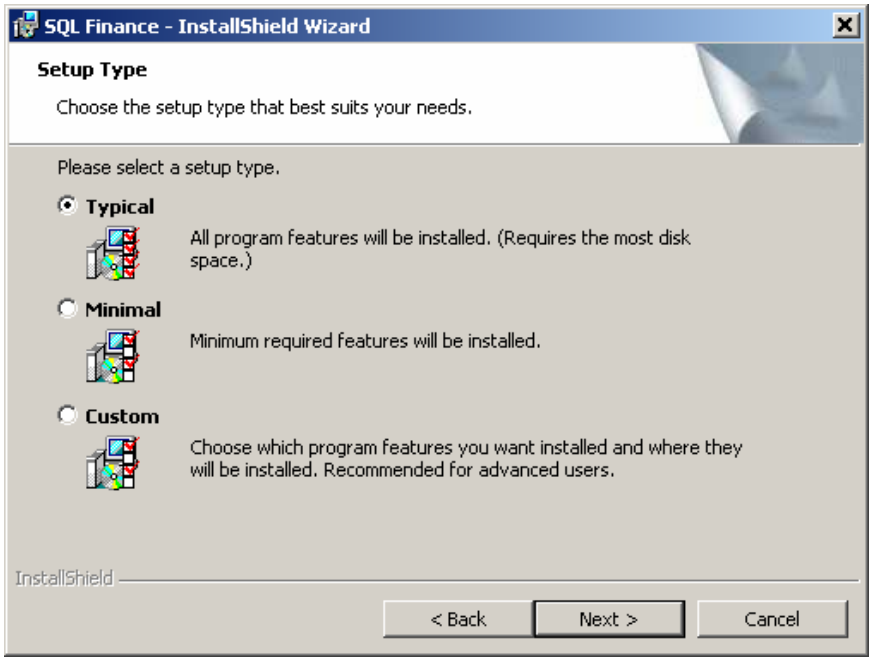
Notes for the use of foreign languages

If you would like to use sqlFinance in connection with foreign languages, you can use the "**BT Language Manager**". A current version of the **"BT Language Manager"** is available at the **download center** of the **sqlFinance homepage**. The **Language Manager** should be installed directly to the root directory **"c:\"**. It includes **tools** for the translation in **any other language** and assures a consistent language use. **sqlFinance** establishes a **connection** to the **Language Manager** with use of the **„lmroot"** variable in the **„l.ini " file** (found in the root directory of **sqlFinance**).

If the **Language Manager** is correctly installed, **sqlFinance** will begin **automatically** to work in the desired **foreign language** (according to the setting **lmroot variable** in the **sqlFinance** application and the **setup parameters** of the active language in the **Language Manager**). Please consider that you must **start sqlFinance twice** in order to get the translation into the foreign language fully active (once in order to change the settings and a second time to get the translation running).

Additional information

On our homepage **http://www.sqlfinance.com** there is always **news *available*** about our company and our software **sqlFinance**.



the sqlFinance installation tool

2. Start with a new mandate

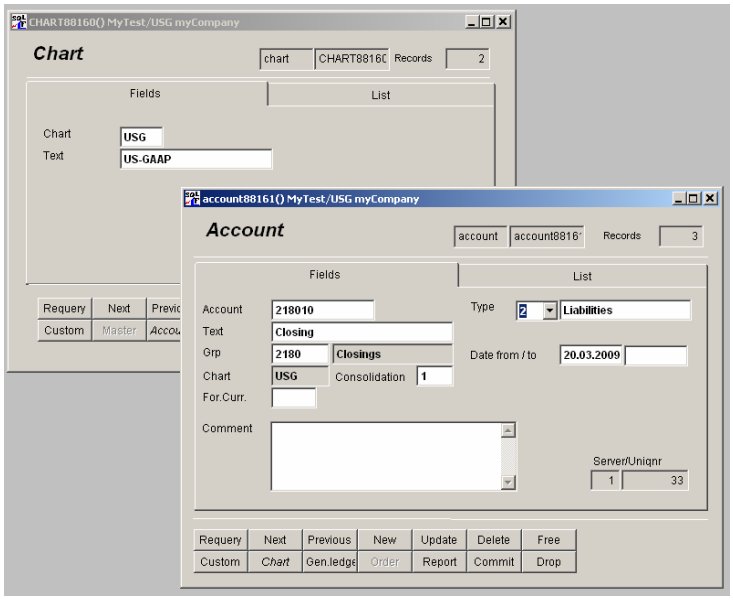
After the installation and the start of the software, the program will immediately establish a **connection** to the **sample database „myComapny"**, select **„myCompany** as the **active mandate** " and then present the **selection tool** on the screen. Now the user can choose **either** to **change the sample mandate** to his requirements **or** to **start entering a new mandate**, possibly with a **new chart of accounts**, or even a **new database** (for the copying of a database see the section **"data management"**).

If just the current mandate **„myCompany"** is to be changed, **delete the given batches** of the sample and start right away with the **entry of new batches**. Note that when batches are erased, the **bookings** of these batches will get **erased as well** (after **confirmation** in the user dialogue with **"yes"**).

If you prefer to **start a new mandate**, proceed as follows:

a. In the sample mandate **„myCompany",** and using the **selection tool**, select the **chart of accounts**. Enter a **new chart of accounts** with the key **„add",** (e.g. **„USG"** for „US-GAAP") and **confirm** with the **„update"** key.

b. As a **detail** to the chart of accounts, select a form with **accounts** (click on **the "detail"** key, third from left, second row of form keys) and **enter** a number of **accounts**. The accounts must be **attributed to groups**. If necessary you can **select** (with the selection tool) all groups and **add new groups** at your convenience. For each account enter the **type** as well (**1 assets, 2 liabilities, 3 expenses, 4 revenues**).



The new chart of account „USG" and the closing account „218010"

c. The new **chart of accounts** should include the following two **special accounts** (preferably placed in the liabilities):

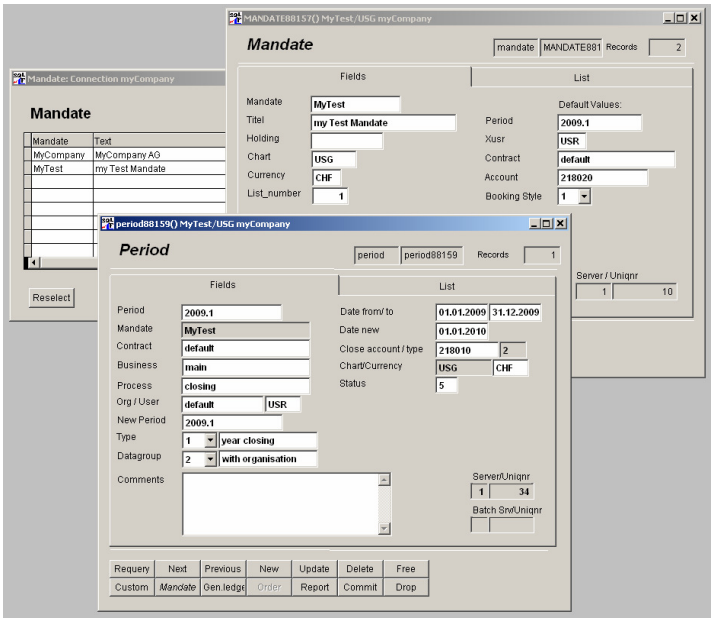
- closing account
- default batch account

d. Using the selection tool, select the old mandate **(„myCompany")**. Enter a **new mandate** (e.g. **„myTest"**). In the new mandate, set the **chart** value to the new chart of accounts **„USG"**, enter the value for the **closing account, but not yet a value for the period.**

e. After the entry of the new mandate you are asked in a user dialogue, if you want to **accept this mandate** as the **current mandate**. Answer with **"yes"** and get (automatically) to the **mandate tool**, where you confirm your current mandate setting with the key **"continue"** and return to the selection tool. Now you can **reselect the new mandate**, possibly using the keys **"next"** and **"previous"** to scroll from the old to the new mandate.

f. With a double click on the field **„period"** of the mandate form (setting the field value to the wildcard **„%"** to get **all records** of the table), call up the **period form** and a **new period** (e.g. **„2009.1"**). The field for the **closing period** („new period") should be **left empty** for the present. **First** enter a number of **periods** with **empty closing period values** and **then adjust the closing period**.

g. Through scrolling get to the desired **period.** Quit the period form using the key **„quit"**. At this moment the value of the **selected period** is **transferred to the mandate form**. Confirm this change in the mandate form with the **"update"** key. Now the mandate has **all default values set** for the current period and the default account of the batches.



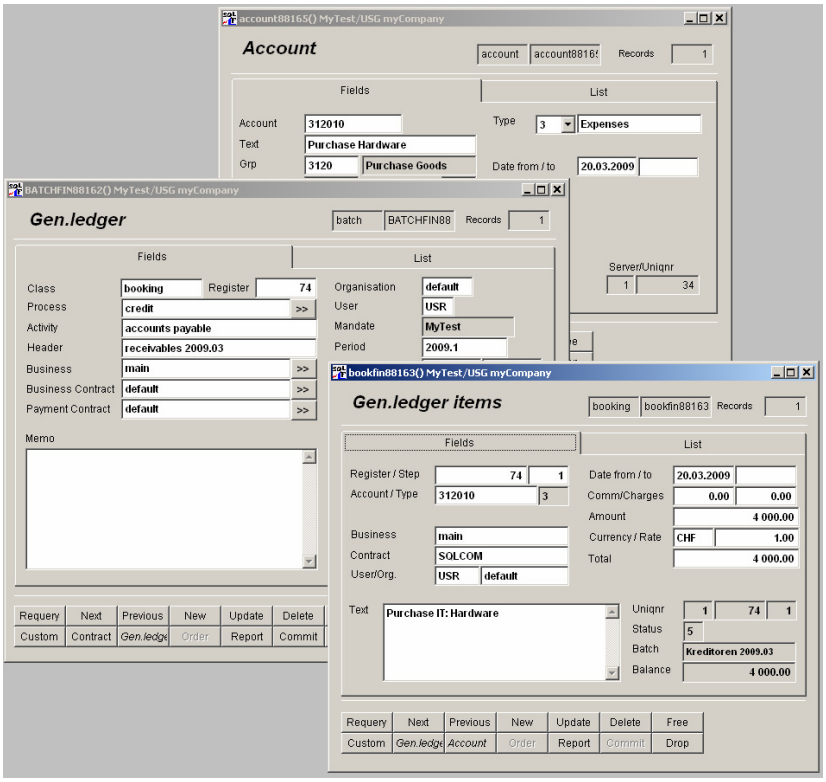
Forms „mandate" and „period" with mandate tool (top left) for the activation of the new mandate

h. Select a **batch form**. In the empty form, **enter a first batch**. The form **requires** an entry for the fields **„process"** and **„activity"**. For the lookup of process entries, set a wildcard (percent sign **„%"**) in the **„process"** field of the batch and use a **double click**. Now, in the **process form**, select a **process** or enter a new process. Get **back to the batch** with the key **„quit"** or the key combination **„ctrl-F9"**: the values for **„process"** und **„activity"** will be **transferred**.

i. **Select** the bookings belonging to this batch with the key **„detail"** (or **„gen.ledger bookings"** respectively, third key from the left in the second row of form keys). In the **empty form**, enter the **first booking** with the desired value for the fields account and amount/total.

j. Entering the value **„SQLCOM"** in the **„contract"** field, consider how the „**whichpen"**-function retrieves the full value through a **lookup feature** and copies the default values for business, organisation, account and amount from the **contract** to the **booking**. It may be necessary to go through the contracts and **enter new records** or **adust the default values** to get the correct settings for the new chart of accounts and the business and organisation values of your specific environment.

k. After the entry of the first bookings, check the **balance report** to see **how the entry reappears in the listing**. Select the **current period**, call up the form **"reports"** (fifth key in the second row) and select a report (like **"details per account"** or **"balance (totals)"**. Start the report with a **double click** on the list item **"preview"** or the "print" key.



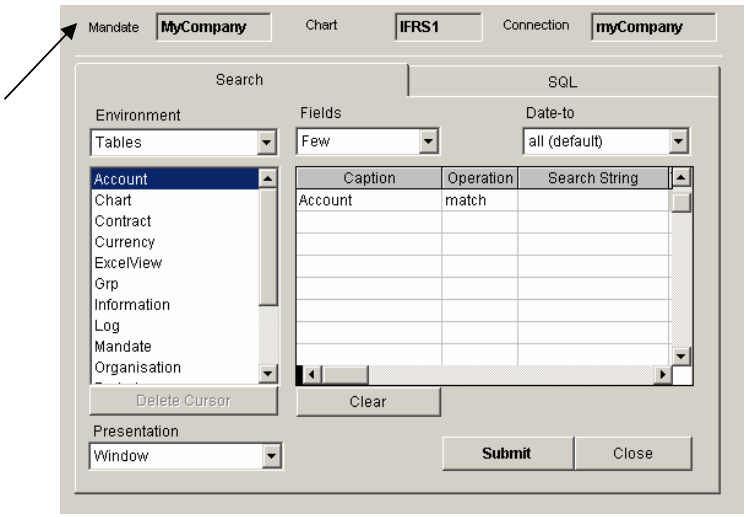
Forms for general ledger batch with gen.ledger bookings and the account "purchase raw material"

l. Before closing the period **commit the batches** (key **„commit"**). This will **verify the total of zero over all bookings in the batch**.

m. For testing it is most convenient to use the default values **„test"** for the **process** and **„default"** for the **contract** field.

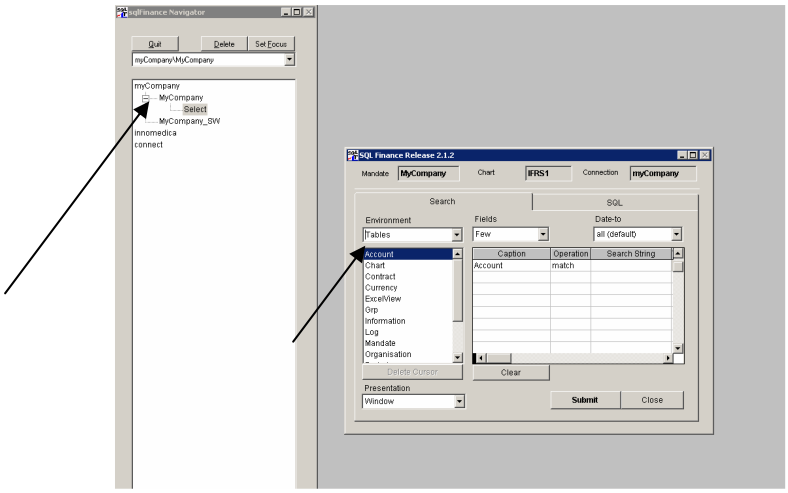
3. Application Overview

The application starts with a **login procedure** that leads to a **selection window**, A **unlimited number of forms** can be opened. Each form will give access to a **table** and offer either a **list view** with several records in a grid or **a field view** with field headers and values. With the **"master"** and **"detail"** buttons you can open **further forms,** referring to the current record data of your initial form.

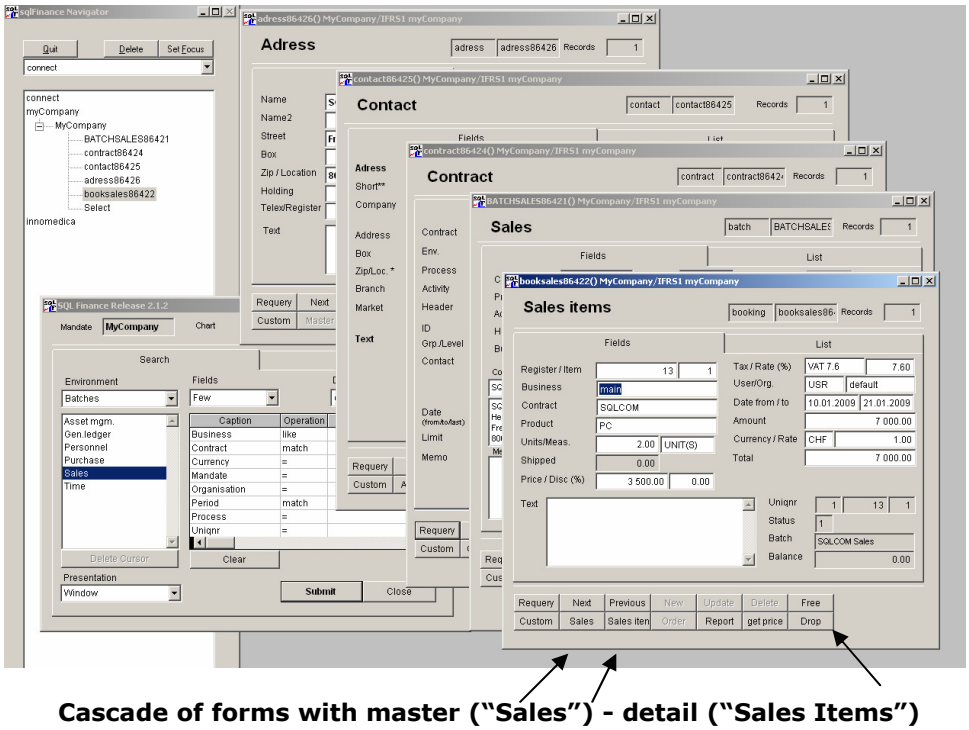


Selection tool within the mandate "myCompany"

Several in such a manner connected forms are organised in a **cascade**. With simple clicks on the "**master"** and **"detail"** buttons the user can move up and down in the cascade. In the cascade only the **currently active form** is visible. With **several open cascades** (each one again with several forms) the application remains **clearly structured** and **easy to handle**. When desired, you can **free a form** from the cascade (button "release"/"free", on the right side of the upper key row) and then open further forms in a **new cascade** using the **"master"** or **"detail"** buttons respectively.



Selection tool with a treeview

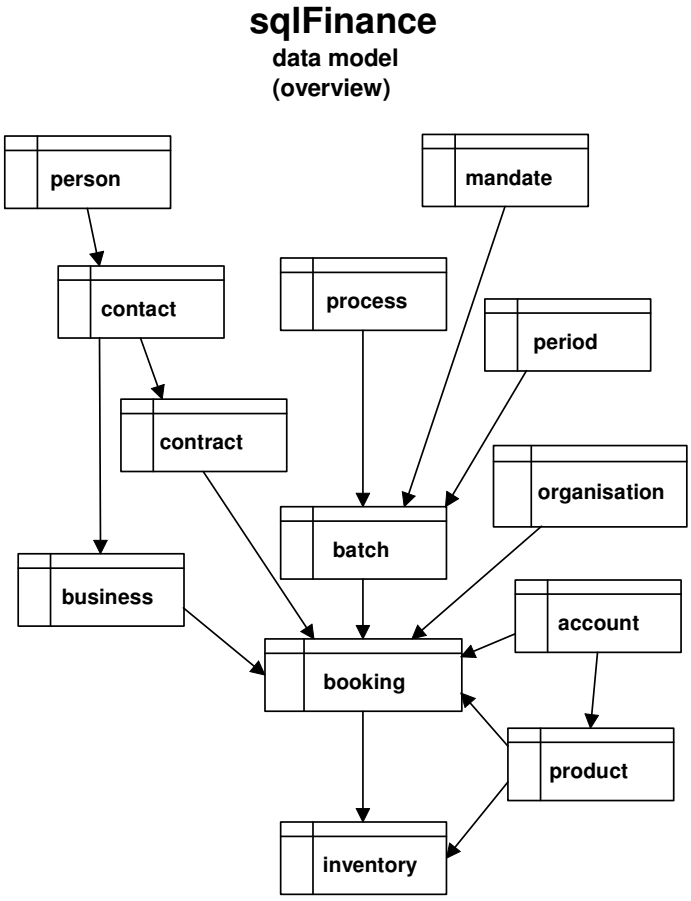


Cascade of forms with master ("Sales") - detail ("Sales Items") and release button

Use the window with the **tree structure** in the left half of the screen (see illustration above) to overview and handle **several groups of forms** (cascades). Each form of a cascade can be **made visible** by clicking the **right mouse button** in the tree structure. For the **later retrieval** of selected data, forms can be pulled over a **depository tool** ("records" tool), in order to **store** a specific data selection.. To the depository entry a comment can be attached. The saved forms and their data selection can later be **recalled** with a double click on the "select" key of the "records" tool.

By "**drag and drop**" with the mouse data sets can be copied from one form to another. With this technique you can **copy bookings** from a one period/mandate to another period/mandate. A **repeat function** makes it possible to **copy entire booking lists**.

The form cascade follows the hierarchical structure of the **data model**. For the user's convenience, the model was kept as simple as possible. The main elements of the model are shown in the flow chart below:



In sqlFinance, the data model drives the cascade order of the forms

The data model's core element is the **booking line**, to which many reference tables (e.g. currency, account, organisation etc.) correspond. In **mathematical terms**, a bookkeeping system can be considered as a **multidimensional structure**, in which different **parameters** – with the amount as the **resultant** - are directed to the **booking line**.

1. Booking line

The booking line contains basic values and parameters. **Basic values** are simple entries in a field. Their value is not verified by the system. **Parameters** however refer to an entry in the appropriate **reference table**. For example, an account value – in order to be accepted in a booking record – requires an entry in the account (master) table.

Basic Values
- Text
- Amount
- Line Number
- Date

Parameters
- Period
- Account
- Contract
- Business
- Organisation
- User
- Currency

Throughout the application, **master tables** can be accessed by a **double click** on the **referenced field**. Double clicking in the "account" field of the booking form opens an "account" form with records corresponding to the value entered in the "account" field. The **lookup action** is based on the **value** the user has entered in the reference field. This value can include **wildcards** like "MyCompany%", meaning in this case that the lookup should list all records with a contract name beginning like "MyCompany…". Enter a **search value** in a parameter field and use a double click to select a **master form** with a list of corresponding data. Note that a booking entry is only accepted when **all parameter values entered** lead to **one and only one parent entry** in the respective table.

Based on field entries in the **contract table**, the system can set a **default value** for the **account** and the **organisation**. While the contract name is typed in the contract field of the booking form, a **whichpen feature automatically checks the master contract table**. As soon as it finds only one contract (mostly after three or four characters), it will **auto finish** the contract typing and **copy** the **account and organisation defaults** from the **contract record** into the corresponding fields of the **booking form**. The cursor then is refocused at the amount field.

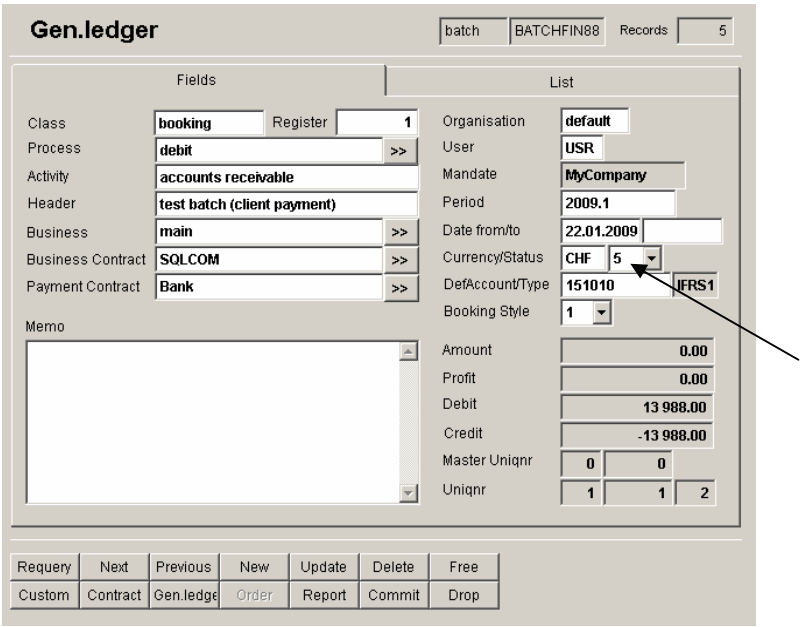
Through **limitation of the typing** to few characters. the entry of accounts receivable and payable transactions can be made **very efficient**. Through the contract's default values a **consistent assignment** of **account** and **organisation values** is assured**.** During the entry of booking data, the user doesn't need to lookup any values for verification. So, the **typing is limited to a few characters** for the identification of the contract name and entry of the amount.

2. Batch

Basic Values
- Reference
- Header
- Text
- Date
- Status

Parameters
- Mandate
- Period
- Process
- Contract
- Business
- Organisation
- User

A number of **booking records** are grouped in a **batch.** For **new bookings** entered, the system will use the **batch defaults** (contract, organisation, user, date). Later, when a contract value is entered manually, the previously described help ("whichpen") feature starts working again and **overwrites** the batch default values with **default values of the contract**. In order to be able to enter new booking records, a **booking window** should always be opened in a **cascade** with the appropriate **batch ("master" entry)**. Otherwise all the (batch) default values have to be entered manually.



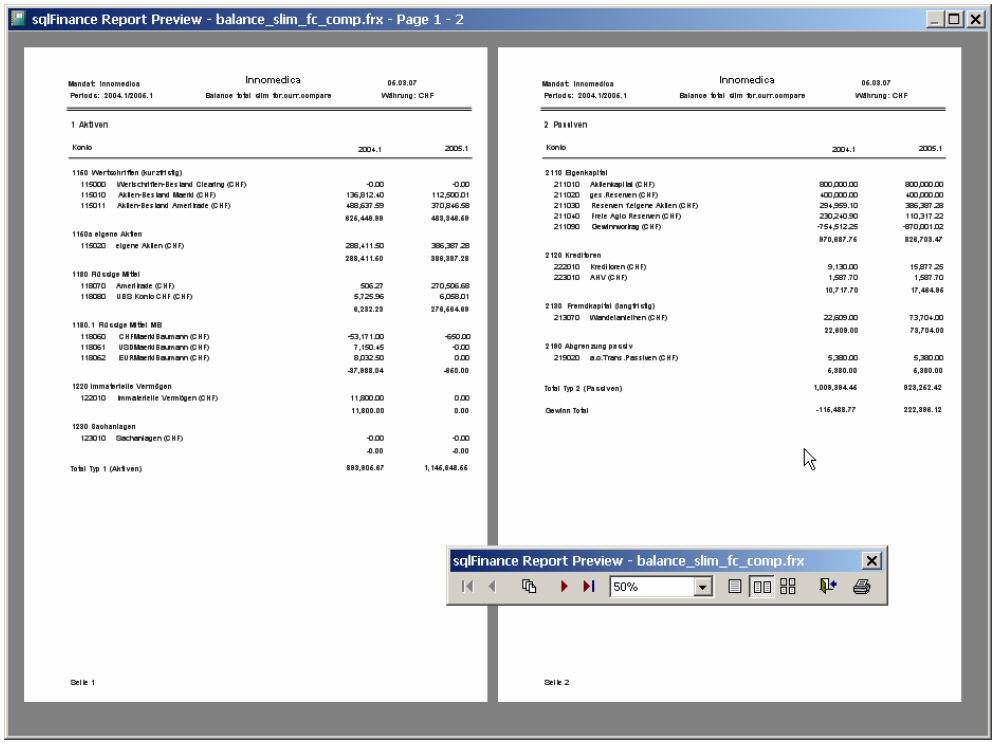
Batch field structure with status set to value 5 (open)

Using the fields **„class", „process", „activity"** and **"header",** batches can **be organised** in a very efficient way. The most important is the field **„process "**. Batches with the **similar booking patterns** are **summarized** in a **process**. While the **process** describes the **general pattern**, the **activity** describes the **specific pattern** of the current batch. For example, if we take the process **„depreciation"**, then the activity could be **„depreciation hardware"** and the

header **„pc's and servers 2009"**. Likewise, to a process **„bank interest and expenses"** would belong a **number of batches**, one batch per bank account**,** The field **"class"** is used to group the processes. In the batch form, while typing characters into the field **„process"**, open the process with a **double click**. The characters entered are used as **search values**. Select a process record in the data list and leave the form with the key **„quit"** or with the hot key **ctrl-F9**. With the return from the process to the batch form the values for the fields **„process"**, **„class"** and **„activity"** are transferred as well. All the user still has to do is to adjust the value for the field „activity" and enter a value in the **„header"** field suitable for the new batch.

3. Close and Commit

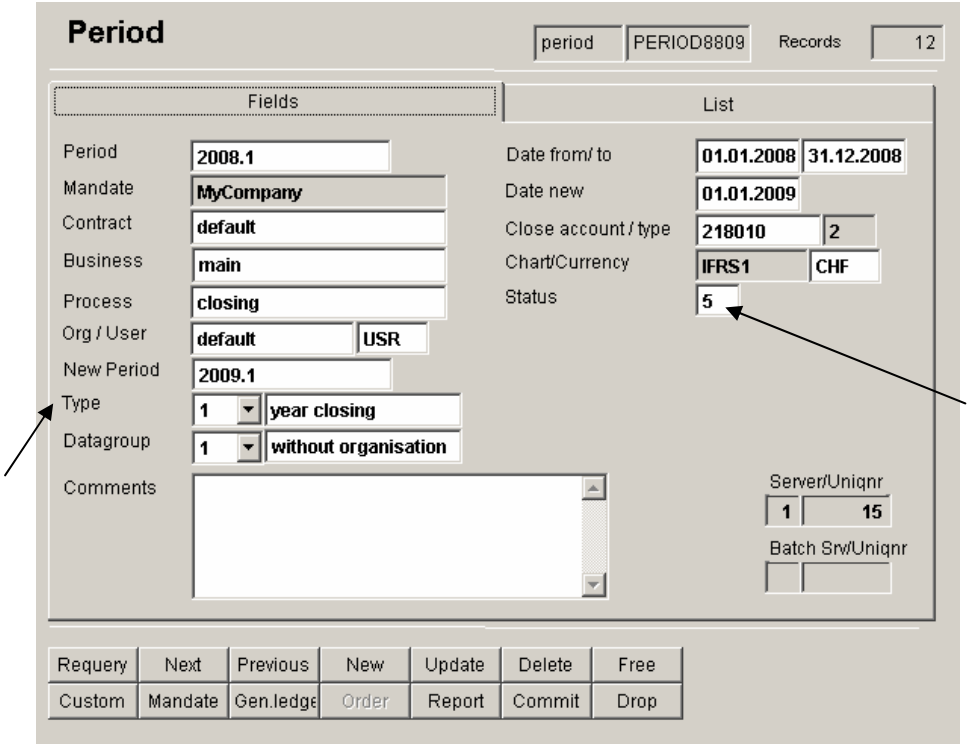
Batches have a **status field**. With the function key **"commit"** a batch can be registered for the **final balance and income statement**. At this time the **sum of all bookings** in the batch must result in a **value of zero**. If this requirement is fulfilled, the status field shows the value „1" (closed). The final **balance and earnings statement** may only include batches with a **status value** of **"1"**. Up to this moment (of closing the year definitively), reports with open and only partially committed batches can be generated as well. With help of the status field you can include or exclude certain batches and generate different reports. This feature is practical during the **closing of the period**. Some batches used for **internal calculations** may finally **never be committed**, whereas for the **revision**, a **final report** based on **only committed batch**es is necessary.



Example of a yearly report with foreign currencies and previous year comparison (with courtesy of Innomedica Holding Inc, Zug)

In a similar way to the commitment of a batch a **period** can be **closed**. In the period window, use the **"commit"** key again to close the period. A batch in the new period with bookings will be generated, each booking with the **total per account** of the old period. Note that the totals will only be calculated over **committed batches** with a status of "1". If you close a **subperiod** into a **main period** (e.g. month-end closing into the current year), totals from the **balance** (types "1" and "2") as well as the **income statement** (types "3" and "4") will be generated. If you close the **main period** into a **new main period** (usually from the previous year to the new year**), only the balance totals** (types "1" and "2") and an entry for the **profit and/or loss** of the period will be generated. The **sum** of these **closing bookings r**esults again to a **total of zero.**

All **commit** and **closing** functions are **reversible**. Closed periods can be reopened and closed batches reactivated. No complicated procedures for save and restore are necessary in this process. The required **consistency** for the **revision** is not obtained through complicated program functions, but rather through a **non redundant data model**, which is mainly based on the **simple cascade (period - batch -booking)**.



Period 2008 with status set to value 5 (open)

4. Contract and business

The **contract** and the **business** fields are the key to the **contact management** (**"Customer Relationship Management - CRM"**). The **CRM** stores detailed information about the **contract partner**, like address, telephone number, civil status, etc. A **contract** or a **business** is always entered as a **detail** to the **contact** (private, business contact). The **contract** registers thereby the **contracting party** to which the **booking line** relates. For **invoices** this is the **customer**, for **purchase** transactions the **supplier**, for **bank** transactions it is

the **bank account**, for **insurance transactions** the **insurance policy**, for a **hirer** the **hiring contract**, for an **employee** the **employment contract**.

The **business** is from a technical point of view built up similarly**,** but rather used for the **summing up** of detail activities (e.g. a number of **customer orders)** to a larger **project.** Since you can capture both the **contracting party** as well as the **business field** on the same booking line, it is possible to get **purchase orders** directly related to a **customer project**. This is important for more **trade oriented companies** and will make it possible to avoid complicated constructs with **project accounts** in the general ledger. In a similar way, **overhead costs** can be assigned to **internal projects** or **business fields**.

5. Product, product line and account assignment

With the application modules **purchase orders**, **sales orders** and **portfolio management** it is possible to enter **product** information. Products can be organized in **item lists (**so-called **bill of materials)**. With the product information included the following booking line results:

Product-related data of the booking line (full version)

Basic Values	**Parameters**
- Units - Price - Discount	- Product (as part of a product line) - Measurement - Tax

Purchase orders, **sales orders**, **portfolio management** and **payroll** use a similar structure for the booking record. As soon as the system finds an entry in the **product field** the **account assignment** is controlled not over defaults of the contract, but over the **product field** and the account assignment information of the **product line**.

When the user confirms a new entry, the program examines the **calculation** of amounts in the booking line, using the formula:

- **units * price - discount = amount.**

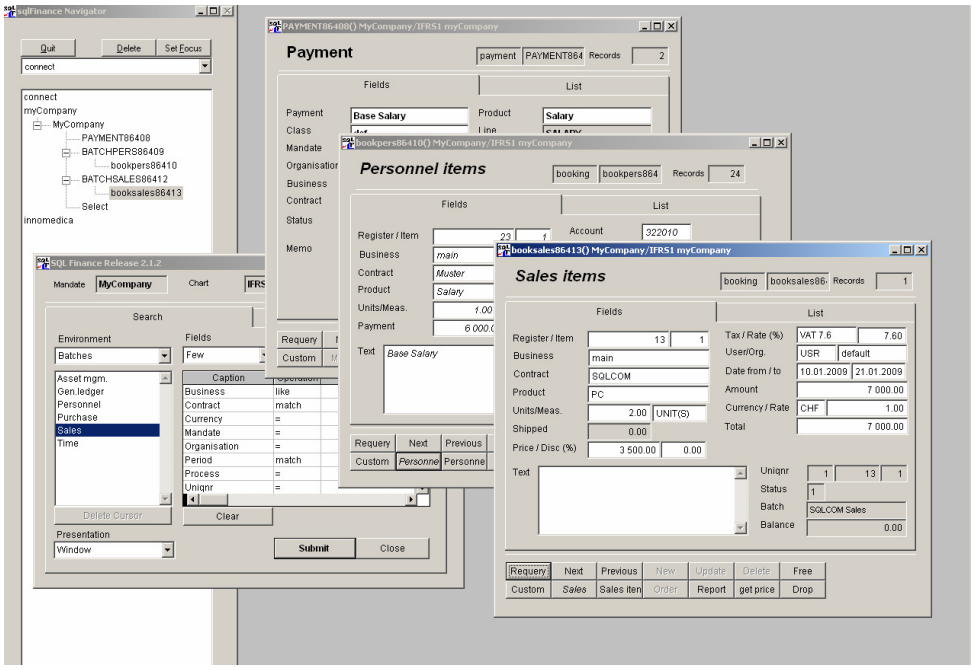
The same examination is done for **foreign currencies**:

- **amount * currency_rate = total**

Note that, if the **total** is varied, the software will adjust rather the **rate** than the **amount**. This is important for (quite frequent) **rounding effects** that normally should affect **neither foreign nor home currency amounts**.

Special fields make it possible to enter **commissions** and **expenses** in the **portfolio management** and in the **general ledger**. If the batch includes

bookings with **taxes,** for each tax an **automatic booking** is added to the batch and therein the **tax total** traced. Different **tax rates** per batch are possible. With this approach **tax obligations** can be examined at **batch level** and be at the same time get **summarized** in an **account** for the **current period**.



Booking examples for portfolio management, sales and the salary statement (full version)

The application structure in the **payroll module** is similar to the sales orders module. Again the reference to a **(service) product** and the **product line** will produce the necessary information for the **account assignment.** Monthly or annual **employee payments** and **social benefit charges** can be stored in **payment** and **transfer tables**. To start a monthly payroll, simply drag a selection of **payments** over a **payroll booking window**. The program will in a first step generate the bookings for the **payments**, in second the bookings for the **social benefits**.

The transfer function for the calculation of social benefits can also be used in the **financial accounting** or in the **sales module** for the generation of **additional bookings**. **Payroll bookings** are preferably booked to a **monthly** (intermediate) **period**, so that in the **general ledger** only the **closing entries** with a **total** per account appear (totals per **cost center/organizational unit** are also possible**)**. This way the general ledger will not include payroll bookings and the details of the salary payments will not be available in queries.

4. Single and double sided bookings

As a result of the consistent application of the data model, **sqlFinance** can offer the use **single sided** and **double sided** bookings **within the same mandate**. In the following part the two methods of **booking data entry** are briefly presented:

Method 1: single sided booking style
Documents with several bookings, such as:

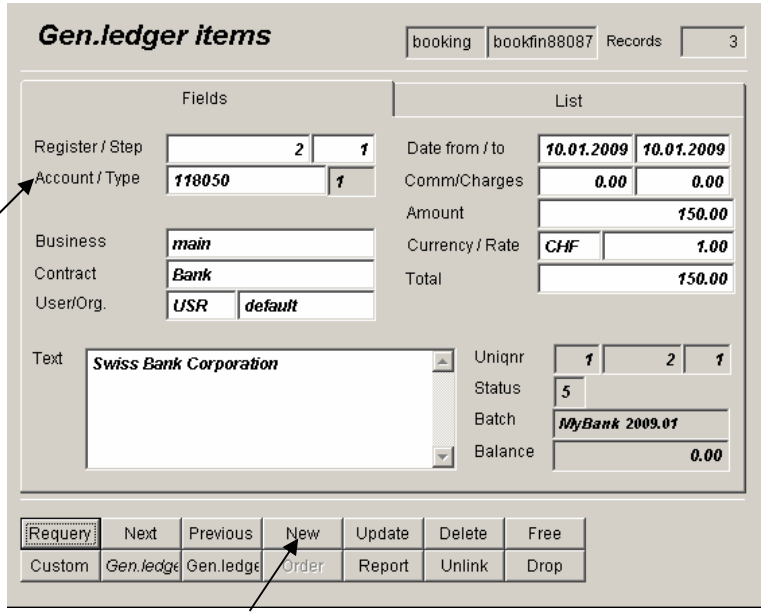
> account, date, text, amount

whereby must be considered: sum (amount1, amount2, amount3…) equals zero.

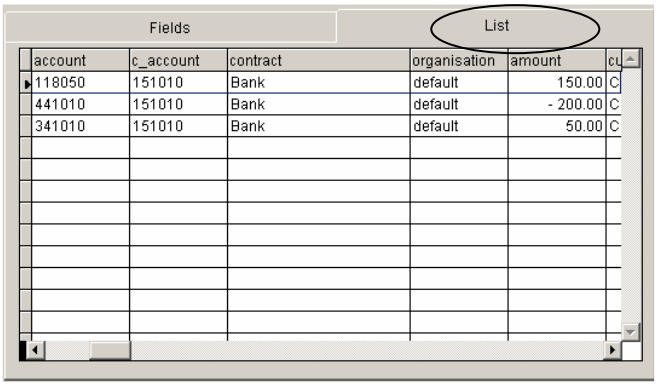
In the report the accounting lines are represented unchanged:

> account1, date, text, amount1
> account2, date, text, amount 2
> account3, date, text, amount 3
etc.

Note that only reports with **all bookings per document** (typically balances and profit/loss statements) will add to a **sum of zero**. Reports of **selected accounts** will in contrast add to the **sum of all movements in these accounts**.



Structure of an accounting line in the financial accounting of sqlFinance (method 1)



Example with three bookings which are summarized in a batch with a total of zero

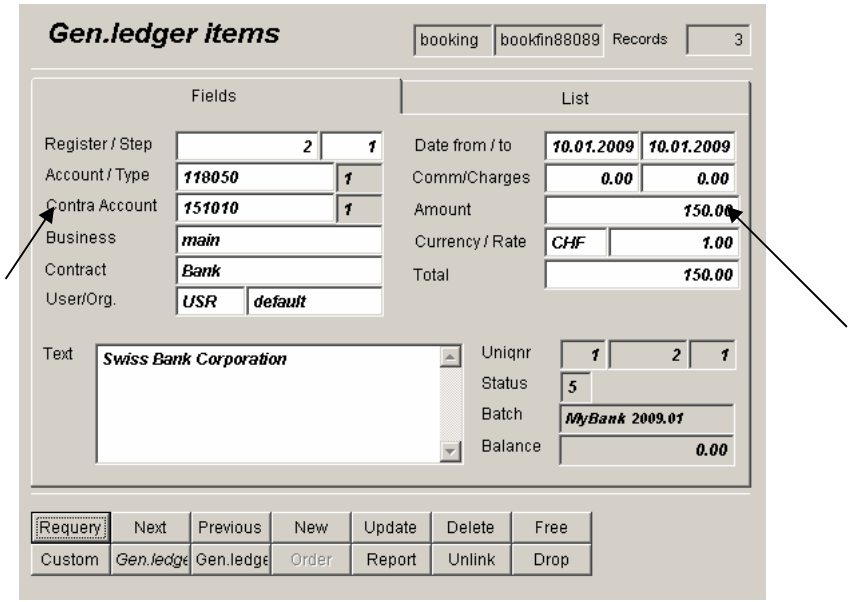
Method 2: Two sided booking style
Individual debit/credit bookings (accounting lines) like:

> debit account, credit account, date, text, amount

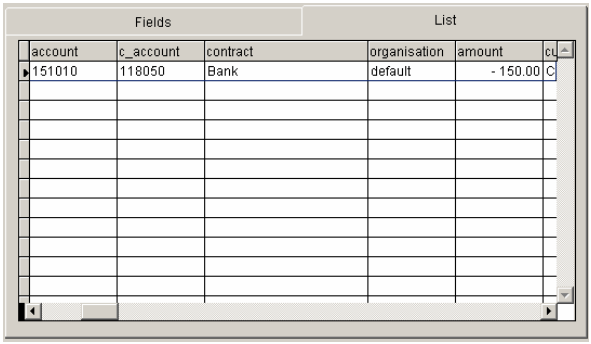
In the report the accounting line is divided into two bookings, which always result to a total of zero, e.g.:

> debit account, date, text, + amount
> credit account, date, text, - amount

With **method (2)** the zero total rule is fulfilled at each accounting line (with debit account, credit account, amount). In the report, this accounting line is split into two bookings with the same positive and negative amount (debit account, + amount and credit account, - amount).



Structure of an accounting line in the financial accounting of sqlFinance (method 2)



Fields List

account	c_account	contract	organisation	amount	cu
151010	118050	Bank	default	- 150.00	C

Example with one single sided booking, which in the report is split into two bookings

In the attempt to keep the bookkeeping as understandable as possible, the software usually **changes the representation** of (2) liabilities and (4) revenues. By reversal of the sign the usually negative numbers are represented positively. This function is used particularly within the reports. If lists are grouped according to the type (1 assets, 2 liabilities, 3 expenses, 4 revenues) the program **changes the representation** of **(2) liabilities** and **(4) revenues** from **negative** to **positive values**. The profit in the balance is often omitted, and the **profit** from the **income statement** (- revenue + expense, usually a negative value) is represented with **reverse sign** as a **positive number**. Simple **booking lists** however will still indicate the bookings with their **mathematically correct value**: **Assets and expense** with usually **positive values**, **liabilities and revenues** with usually **negative values**.

Receivables are to be booked as follows:

(1) + assets
(4) - revenues
= 0

For the assets side can be used (e.g.) a bank, a cash or a receivables account.

For the booking of **payables**, proceed accordingly:

(1) - assets
(3) + expenses
= 0

Instead of booking the expense within the assets (e.g. a bank account), you can also use a liabilities account (e.g. accounts payable). For the expenses or revenues often several accounting lines will be used, like for example a number of purchases to be paid in one single payment, or a number of billing items, that add to one final receivable amount.

Some forms (like purchase, sales, receivables or payables) use presentations with **inverted signs**. While **purchase and accounts receivable** bookings contain anyway usually **positive numbers**, bookings for **sales** and **accounts payable** are usually stored as **negative numbers**, but represented in **the form** as **positive values**. In order to keep a total of zero, usually different bookkeeping forms are used in combination. For example, a **sales batch** will use sales bookings (with reversal) and accounts receivable (without reversal), a **purchase batch** will use purchase bookings (without reversal) and accounts payable (with reversal). To the purchase and sales batches always belongs a

booking form for the **taxes** (e.g. value added tax). The taxable value is usually an obligation, so the tax form uses reverse values.

Booking forms used with sales batches:

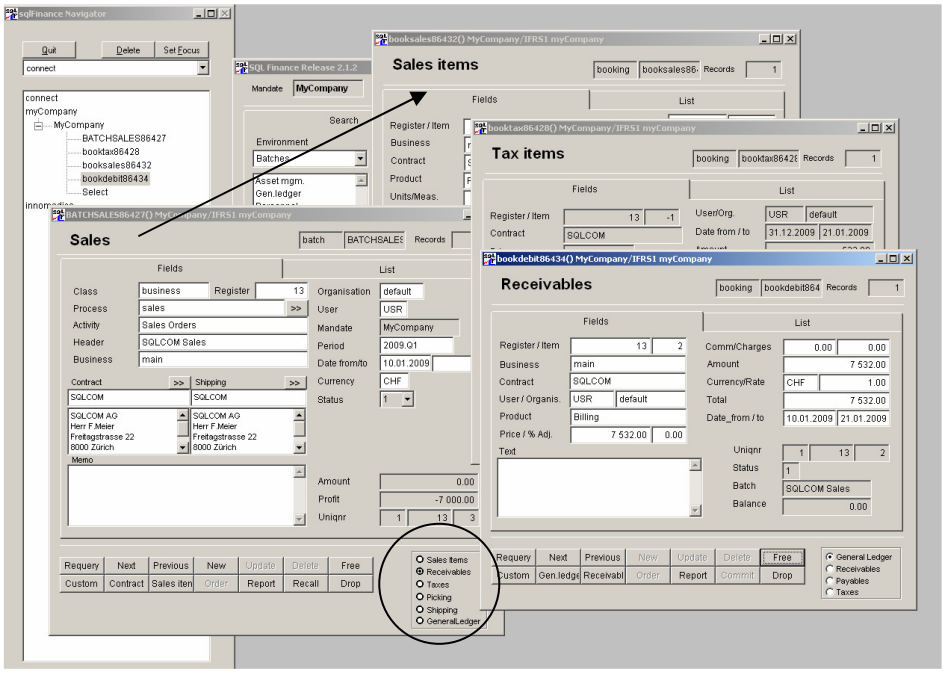
(1)	+ assets:	acc.receivable form	without reversal
(2)	- liabilities:	tax form	with reversal
(4)	- revenues:	sales form	with reversal

Booking forms used with purchase batches:

(2)	- liabilities:	acc.payable form	with reversal
(2)	- liabilities:	tax form	with reversal
(4)	+ expenses:	purchase form	without reversal

Note that for **each line** in the example above, there could be **several bookings**. Multiple earnings or expenses bookings can represent **entire sales or purchase orders**. Even **accounts receivable or payable entries** may include several bookings for **partial payments** or **payment adjustments**.

The system will automatically track **tax liabilities**. More than one tax type per batch is possible (leading to more than one tax booking per batch). In VAT countries, purchase and sales taxes can be directed to a clearing account, leading to valid report usable for the **tax declaration**.



Example of a sales order batch (at left) with booking forms for receivables, taxes and sales items (full version)

The different **booking forms** (e.g. accounts receivable, sales and taxes) can be called from the **master batch**. Choose the booking form in the **selection pad** at the right bottom end of the batch form and then activate it with a **double click** on the mouse (positioned over the selection pad or over the "detail" key.

5. Data Management

Before you start working with **sqlFinance** you should first organize your **master data**. To the master data belong (among other things) the mandate, the chart of accounts, the period structure, the various contracts and the organization.

The organization of the **master data** is to be aligned with the **business structure**. Define your **internal organisation** (similar to the common „cost centers") as well as **contracts** for your customers, suppliers, financial business partners like banks and insurances, and your employees. Material and services are to be registered as **products**, grouped into **product lines**. The **accounts assignment** can be controlled in the **general ledger** through a **default in the contract**, in the **purchase and sales order** through the **product line**. The **payroll** will use the product as payment classification and refer for the account assignment to the **product line** as well.

Find hereafter a **short overview** of the **master data**.

a. Connection

As soon as you open **sqlFinance** you have to establish first a **connection** to the **target database.** For this purpose use the **connection tool**. You can either open a connection to a **local data base** (ODBC type set to **"local FoxPro"**) like in the connection **"myCompany"** for the sample database "myCompany", or to a **server data base** (ODBC type set to **"SQLServer"**). In the connection tool **sqlFinance** specific information about the **physical database** and the **ODBC connection** is saved.

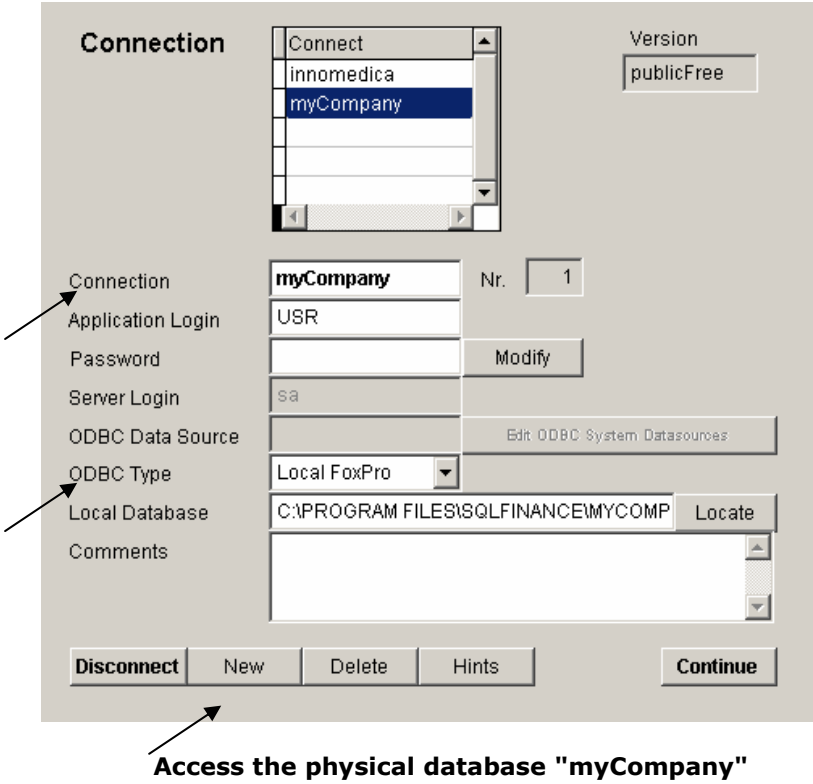
If you open **sqlFinance** for the first time, you will get connected to the sample database **"myCompany"** (in the subdirectory **"myCompany"** of your sqlFinance installation). In the connection tool, with a click on the key **"connect"** you can open a connection to the database **"myCompany"** With the button **"continue"** you get to the next form, the **mandates** tool. Choose the mandate **"myCompany"** and click again on the button **"continue"** to get to the **selection tool,** the main search form of sqlFinance.

Add and connect to a new database

In order to make a connection to a **new local data base**, copy the database **"myCompany"** into a new directory. You will find the **"myCompany"** database in the **sqlFinance** install directory of your computer (e.g. **"c:\Program Files\sqlFinance"**). You may rename the directory (e.g. "yourCompany"). In the connection tool, click on the button **"new"** to start entering a new connection. You can now enter the **name** of the connection and the **target directory** ("local database"). Set the **ODBC type** to **"local FoxPro"**. Un unlimited number of databases is possible. Likewise the number of mandates per database is unlimited as well. Note that for the database **"myCompany"** the pre-set value for the **"user login"** is **"USR"**. An entry for this user **must exist** in the table **"usr"**, with the table "privileges" regulating access to all tables and reports. In these tables **further users** with possibly **reduced authorizations** can be defined.

Databases like **"myCompany"** can be **renamed** at any time by changing the file names "myCompany.dbc", "myCompany.dct" and "myCompany.dcx" in the target database directory. The directory can be renamed as explained above, or the database be copied into another directory in any place on the workstation or in the network. After renaming you also have to change the **connection** or

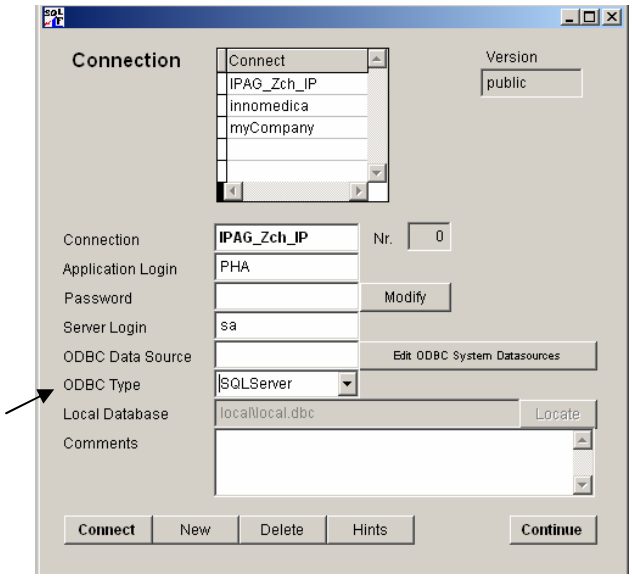
establish a new one. Working with local databases, no further tools are needed for the handling of the connection.



Access the physical database "myCompany" with the "Connection tool"

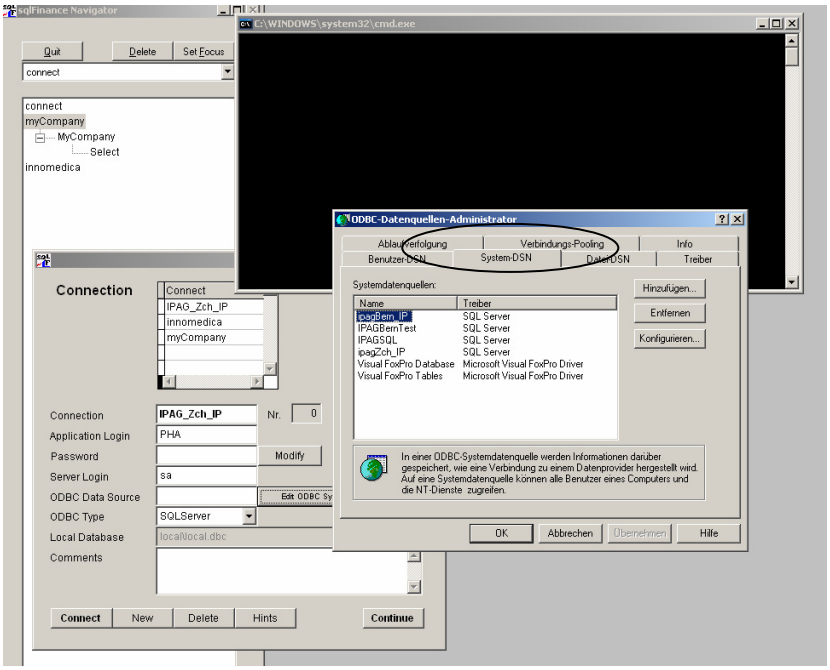
For the access to the server data base **sqlServer** however an entry on the **operating system level** must be provided as well. The server can be positioned in the **local area network** or in addition **remotely** connected over the **Internet**.

The system parameters and **meta data** of **sqlFinance** are stored in the **"sqlf_db"** database in the **"data"** directory. In this database you can define your **own tables and fields**. Efficient the **tools** for the **administration** of these **meta data** are available.



Connection tool to the Server data base "IPAG_Zürich"

In order to make a connection to a **server data base**, set the value of the field **ODBC type** to **"SQLServer"**. Click on the button **"Edit ODBC System Data Sources"** and select in the ODBC manager **"user DSN"** or **"system DSN"**. Enter new a connection with the Driver set to **"SQL server"** with **"add"** and click on **"finish"**. Now you can enter the freely selectable **name of the** connection. Note that the name in the **connection tool** must **identical**. Configure now this connection and identify the correct **server** and the correct **data base**. Note that the database must exist on the server. To test the connection, use the **"connection test"** function (located at the end of the dialog). If a **connection** is possible, the feedback will be "**successful"**.



"Edit ODBC System Data Sources"
(started from the selection tool)

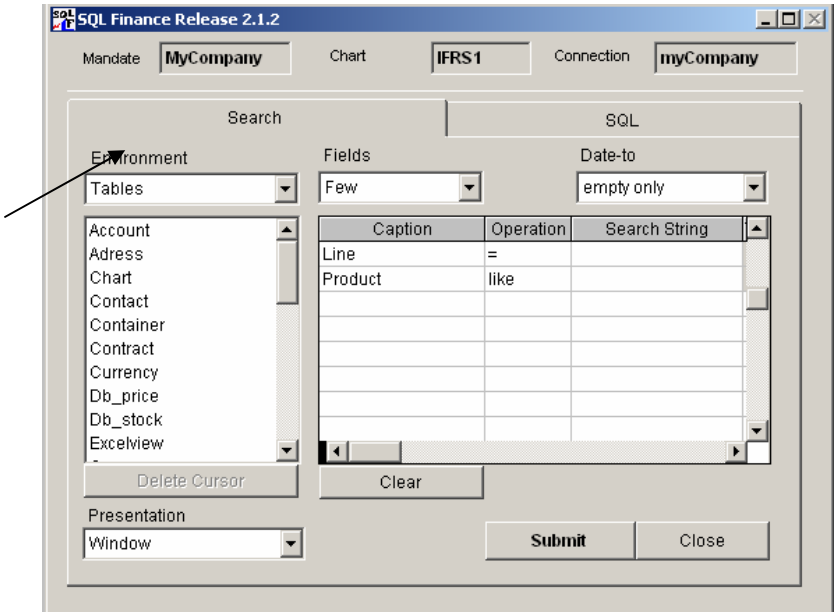
As soon as a valid **ODBC connection** is present, enter the name of the connection in the field **"ODBC connection"** in the connection tool. Use the name of the user for the login in the **operating system** in the field **"server login**" (e.g. **"sa"** for the **system administrator**), in order to be accepted in the **access control of the server database**. In the **field "application Login"** you can identify yourself concerning your **rights of access** within the **application functions of sqlFinance,** for example with your personal or company name. Through the entry of a **password** you can secure the access **exclusively**.

It is possible to have **several** connections open at the **same time** and for example **transfer data** over different connections from one database to another. As soon as you click on a **form**, the **connection** belonging to it is activated. With **drag and drop** you can easily **copy** data of a form to another, even if the forms belong to two different connections. This can be an effective procedure to transmit data from a data base to another one.

If by mistake you delete an existing connection in the connection tool, press the button "new" and enter the name of the deleted connection. As soon as you click on "connect", the system takes over the "old" data.

b. Selection tool

After opening a **connection** and the selecting of the current **mandate**, you get to the **selection tool**, an important aid for the **search of data** in a data base like for example "myCompany". The selection window can be called from the **menu** (DATA/search) as well or by **right clicking** on the position **„select "**in the **treeview**.



The selection tool of the sqlFinance full version

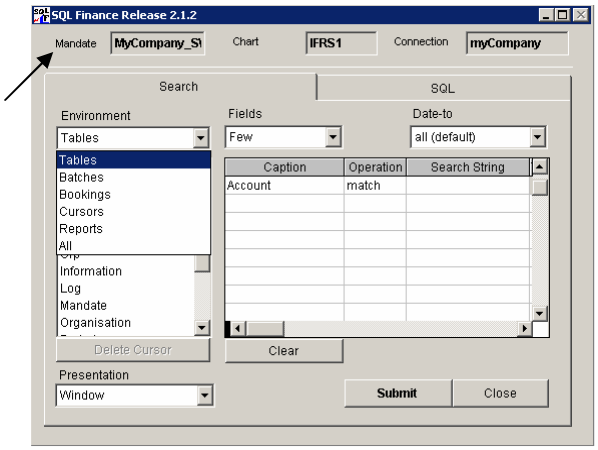
The **selection tool** offers a **first access** to an active data base and enables the user to execute a **query** according to his **search criteria**. Using the search values entered the selection tool will generate an **SQL statement** (Standard Query Language call) and execute it. The retrieved data is read into a **cursor**,

stored in the **flash memory** of the workstation and presented in a **standard form** on the user's screen.

From such an **initial form** the user can start building a **cascade** of additional forms. Using **master** and **detail buttons** on the form or through **double clicks** in certain fields the cascade can be expanded. Advanced users can **modify the SQL statement** on the second page of the pageframe in the **selection tool** (title: **„SQL"**). In the pageframe **order** of the selected data records can be modified as well.

1. Table selection

The table to be queried can be selected in the "selection tool". According to the setting of the upper listbox ("environment") the lower listbox will show a selection of available tables (depending on the "full" or "public free" version and the access privileges of the user):



Selection tool with the table options

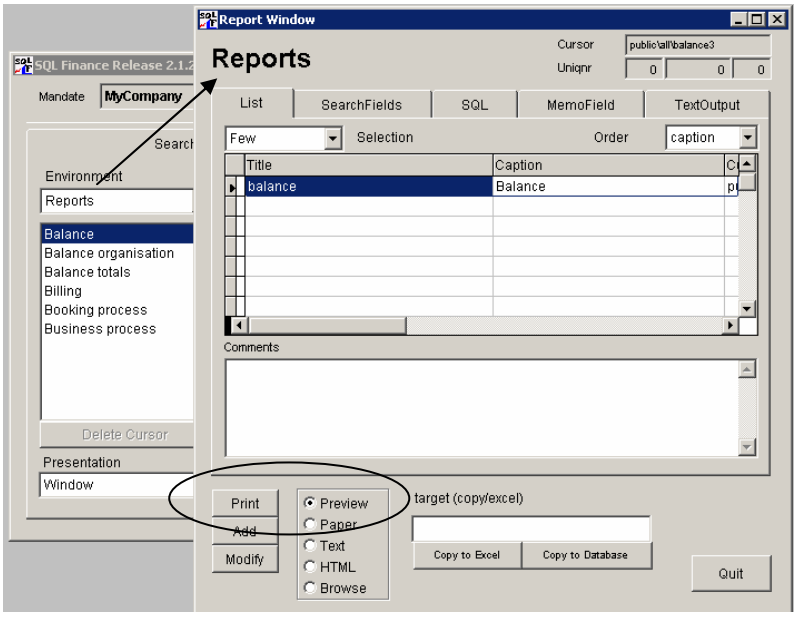
- **Tables**
  A selection of available tables for the **contact management** ("address", "person", "contact", "position"), for the **logistics** ("container", "property", "product", "inventory", "measurement") and for a more **general use** ("process", "price", "tax", "transfer", "payment"); these tables are all mainly **parameters** and their **descriptive supplements** (e.g. the products and their prices).

- **Batches**
  Application specific **batch views** for the **general ledger** and the modules **purchase, sales, personnel, portfolio** and **time management.**

- B**ookings**
  Direct access to **bookings forms** (booking forms are usually selected as detail of a batch form). The master batch form can be activated from the booking form; besides general ledger bookings there are bookings from the modules "receivables", "payables", "purchase", "sales", "inventory", "shipping", "personnel/payroll", "production", "tax", "portfolio management" and "time management".

- **Cursors**
  Selected data is kept in cursors. **Cursors** may be **stored and reselected at later time** (not active in current versions, replaced with the "records" tool)

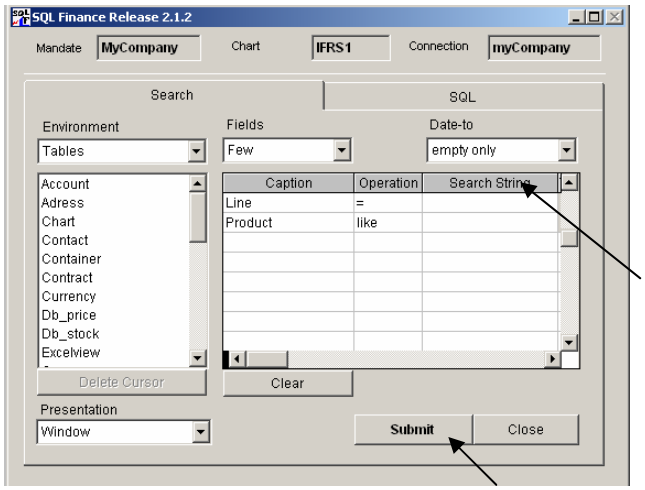
  R**eports**
  Preferably reports are selected from a form, using the reference to a current record (ex. a period, an account or a client contract). In some cases however, the reference to the current record may not be desirable (e.g. a selection over many periods and many contracts). In this case the report can be started from the selection tool. All data of the current database (over all mandates) will be selected, under the condition that they satisfy the query criteria entered by the user.



left: selection tool, right: report window (a doubleclick on line "balance" in the selection tool opens the report window)

2. Search values

When the table to be queried is selected in the "selection tool", the **grid** to its right shows a list of **search fields** for this table with a column for the entry of **search values**. Several search values can be used in a query. The number of fields to be shown in the list can be set in the **"fields" listbox** ("few", "many", "all").



column "search strings" in the selection tool

3. Query execution

With a click on the button **"submit"** the **SQL statement** is generated and executed (e.g. select a company with the name „myCompany" in the address table). According to the value in the listbox **"presentation"** the query will show a **form** or a simple **browse list**.

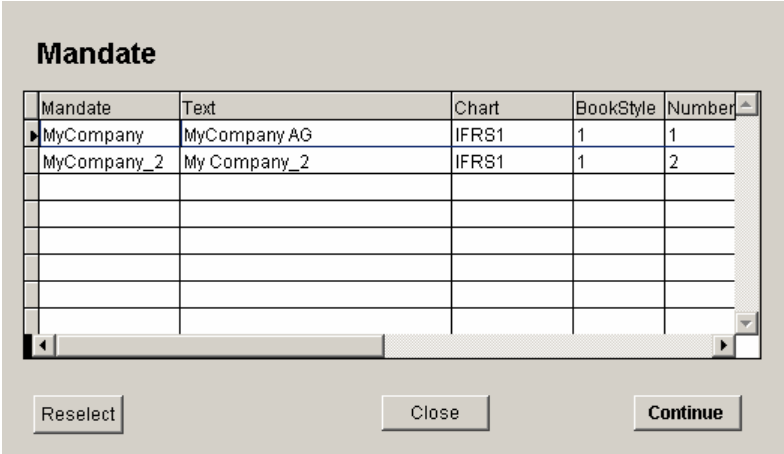
If the query is too long (very rare), it can be **interrupted** with the **ESC** key. If the query is successful, the **application** form is presented and the selection tool hidden in the **cascade**. The next selection step is often the query of **further details** to a current record in the selected list. The selection tool can be **reactivated** with use of the **menu** or a click in the **treeview**.

c. Mandate

As soon as a connection is established, the system activates the **mandates tool**. A Database can include an **unlimited number** of mandates, whereas mandates typically represent **companies in a holding** or any other **entrepreneurial grouping**. Before the selection of data in the selection tool you have to **choose the current mandate** in the list presented in the grid. Then you click on the button **"continue"** to leave the **"mandates"** form and get to the **selection tool**.

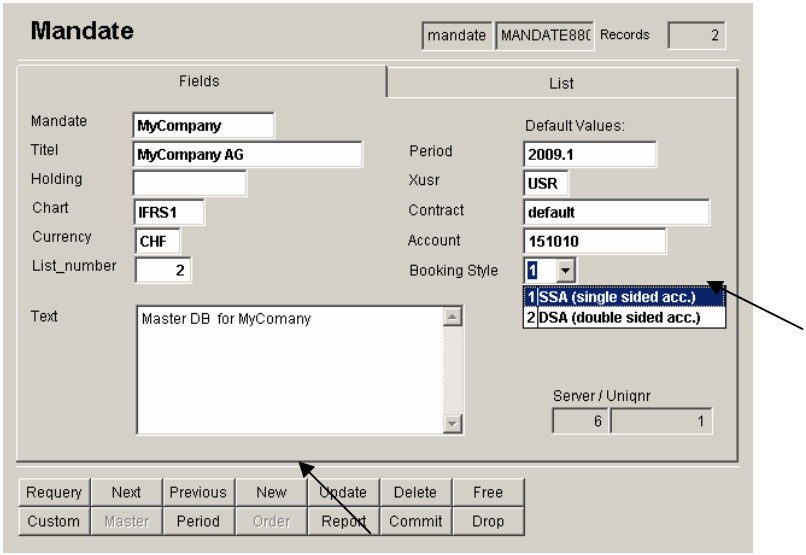
Note that not all tables are **mandate specific**. Data for the **contact management** (CRM customer relationship management) is **not limited** to a single mandate but available over all mandates of the current database. Accordingly, a **database** can include **several mandates**, but only **one single CRM**. If you want to assign a different contact management to each mandate, it is better to organise each mandate in a separate database.

In a similar way, the **chart of accounts** is mandate specific. However, **several charts of accounts** can be registered in a database. The **assignment** of the chart to the mandate is entered through a **default value** in the **mandates form**. It is possible to **change the chart** of the mandate at **any time**. The system can handle **more than one chart per mandate**, but this feature is only recommended for **advanced users** and **special requirements**.



Mandate "myCompany" in the tool "Mandates"

Simultaneous access to **several mandates** is always possible. If required, records can be **dragged** from one form of a **first mandate** over another form of a **second mandate** (using drag and drop). When working with several connections and mandates, it is easier to use the **treeview** (on the left of the screen) and to activate from there a **previously selected form** or the **selection tool**. Note that the selection tool is always set to the **current mandate**, that is the mandate of the **last activated form**. If the is no active form for a mandate, use the **"connection"** and **"mandates"** tools to activate the selection tool with the **correct mandate setting**.



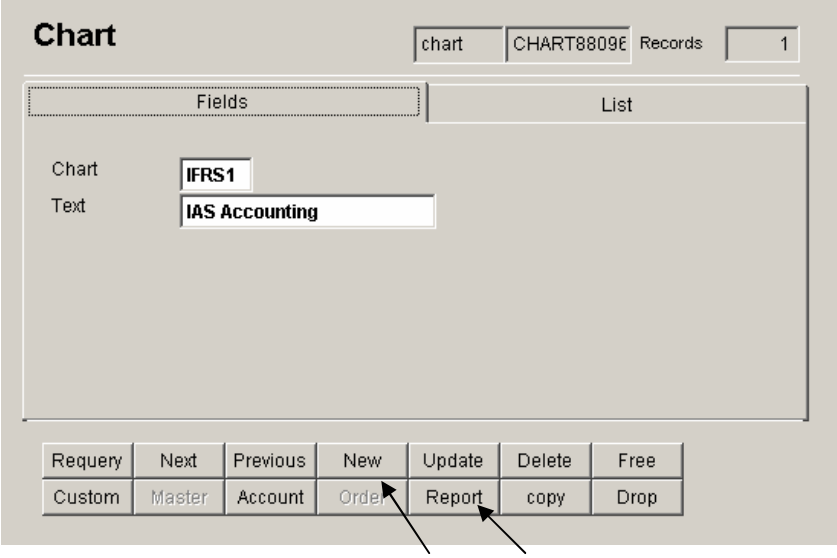
Mandate form with current record "myCompany"

A **new mandate** can be entered in the sqlFinance form (activated from the selection tool: tables, mandate). With a double click on **"mandate"** a form is activated. After a click on the button **"new"** you can enter the new mandate and confirm with the **„update"** key. Then, the software will ask you if you want to consider the new mandate as the **current mandate**. If you answer with **"yes"** the **mandates tool is activated with the new mandate**, and after your **"quit"** click the **selection tool** is presented with the **new mandate** set as the **current mandate**.

d. Chart of accounts

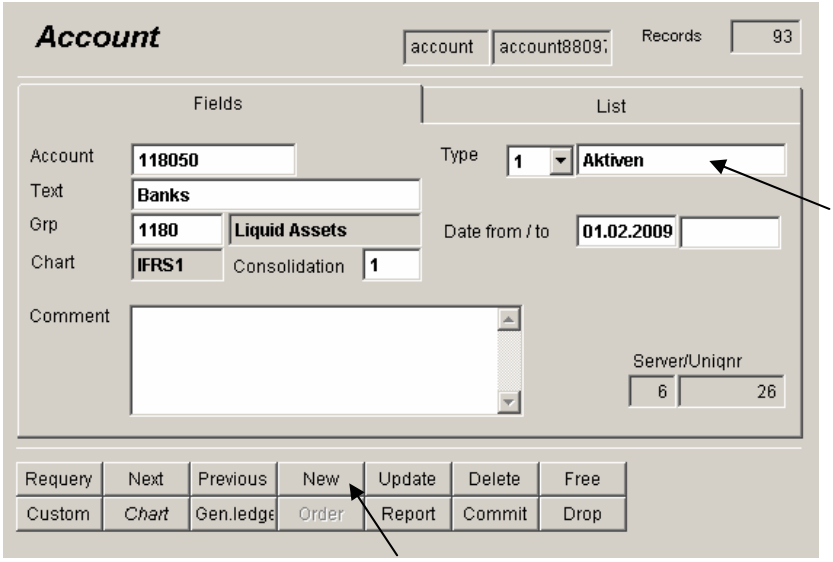
In the database **"myCompany"** an **IFRS** chart of accounts is available. Other charts and account entries can be easily added. In order to enter a **new chart of accounts**, set the **„environment"** field in the selection tool to **„tables"**. With a **double click** on the entry **"chart"** a form for the editing of the chart of accounts is opened. Click on **"new"** to start entering an new chart. With the **detail key "account"** (third key from left in the second line of keys) you can activate the **accounts form** in the cascade and enter new accounts in a similar way.

The **accounts form** can also be activated with the **selection tool**. Again, set the **"environment"** to **"tables"**, and with a double click on the entry **"account"** get a form to enter new records. If you wish, you can open in **another cascade** the **sample chart of accounts "IFRS1"**. View the accounts using a **report** (key **"reports"** and then a double click on **"preview"**). In the accounts form you can see the entered accounts in the **"list"** page (page two of the pageframe). Please remember that after completion of data entry the **chart value** in the **mandate form** must be set to the **new mandate**. Only then the new booking entries will consider the **new chart of account**s. **Preset values** for accounts in the **contract table** may need to be adjusted as well (and in the **line table**, if you use **logistics** or **payroll**).



Form "Chart of accounts"

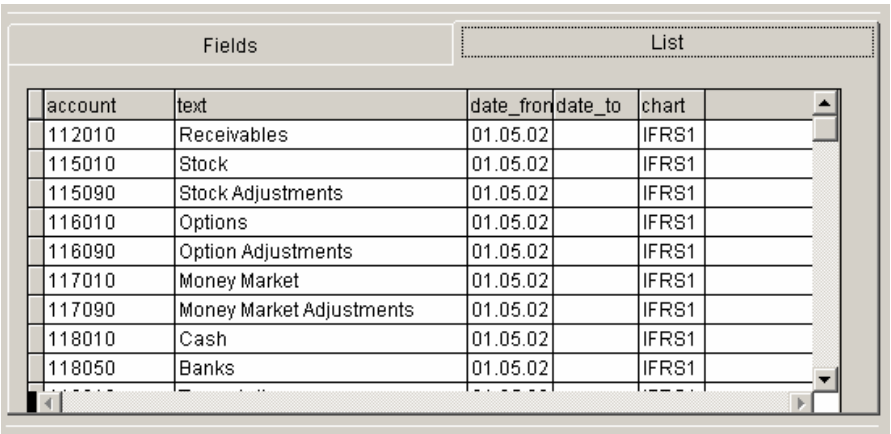
Accounts are structured in **"groups"** and **"types"**. Only the **mathematically** necessary minimum of **four account types** is used (1 assets, 2 liabilities, 3 expenses, 4 earnings). This keeps the application slim and consistent. For other account types like "taxes" and "other income" rather use the grouping structure.



Form "Account"

The **grouping of the accounts** is important for the presentation of data in the reports. Please check the **IFRS chart of accounts** to get an **example** of account grouping. Over the **„groups"** of the accounts you can establish a **multi layer contribution analysis**. The accounts can be attributed to **"parts"** and within these "parts" to a **sorting number**. This way, e.g. a **first layer** can show material purchase and sales, a **second layer** further project expenses, and a **third layer** general administration costs. In a **fourth or fifth layer,** finance expenses and revenues, and tax estimations can be additionally considered. **Reports** for the use of multilayer contribution analysis can be found in the report list of the form **"period"** (click on the key "reports" in the period form).

Every chart of accounts should include a **closing account**. This account is needed for the closing of a period. If the company is profitable, the closing account should be defined as part of the liabilities (type 2, usually in the equity group). In case of losses, the losses might be shown in a closing account of the assets group (type 1).



account	text	date_fron	date_to	chart
112010	Receivables	01.05.02		IFRS1
115010	Stock	01.05.02		IFRS1
115090	Stock Adjustments	01.05.02		IFRS1
116010	Options	01.05.02		IFRS1
116090	Option Adjustments	01.05.02		IFRS1
117010	Money Market	01.05.02		IFRS1
117090	Money Market Adjustments	01.05.02		IFRS1
118010	Cash	01.05.02		IFRS1
118050	Banks	01.05.02		IFRS1

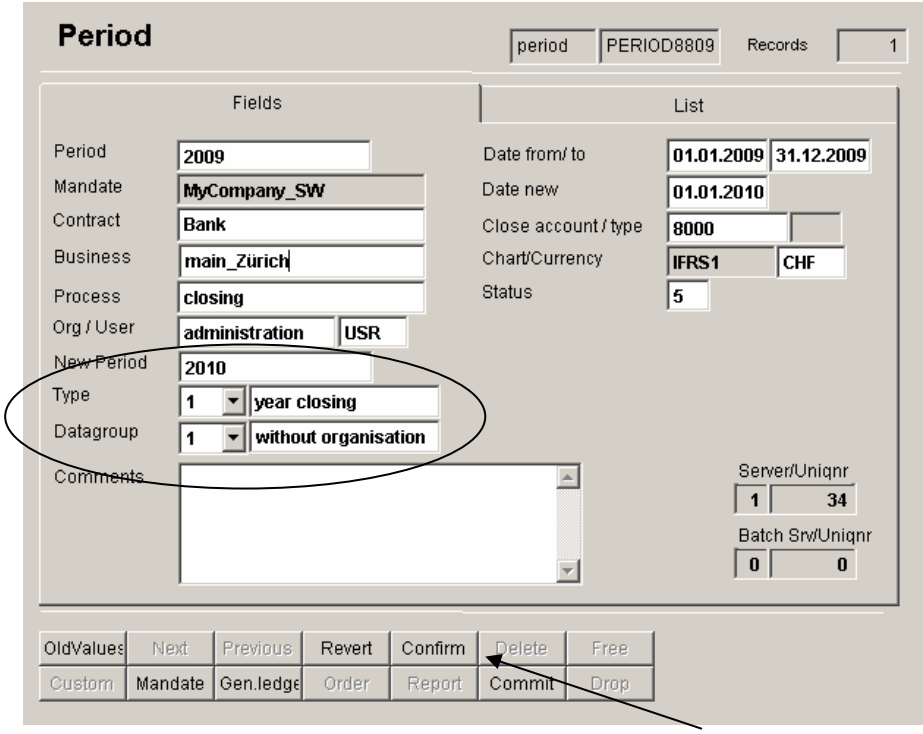
Example of an IFRS chart of accounts

e. Period

Batches and its bookings are normally associated to **periods**. A typical period is the **current year**, lasting from the 1st of January to the 31st of December (other dates are possible). At the period level, a closing account must be entered. When the period is closed, totals per account for the period will be entered as opening bookings of the new period.

Instead of periods, **sub periods** can be used. While **yearly closings** only report **balance totals** for each account (1 assets and 2 liabilities) and a single entry for the **balance profit** to the new period, **sub periods** will report totals for **all four types** (1 assets, 2 liabilities, 3 expenses, 4 earnings) and **no profit/loss entry**. The use of sub periods is not mandatory, but **highly recommended** for **payroll applications**, as well as the handling of high **debit or credit volumes**. Sub periods may be named like 2009.01, 2009.02 etc, periods like 2009.1 etc.

If the **sales or purchase order module** is used, batches can be **sent directly** to general ledger as part of period entered in the sales or purchase batch record. However, it is also possible to **generate a new batch** with **only totals of sales items** (but still the details of the receivables/payables and tax entries). In this case, the period of the new general ledger batch will be set to the master period of the current period (as entered in the sales batch record).

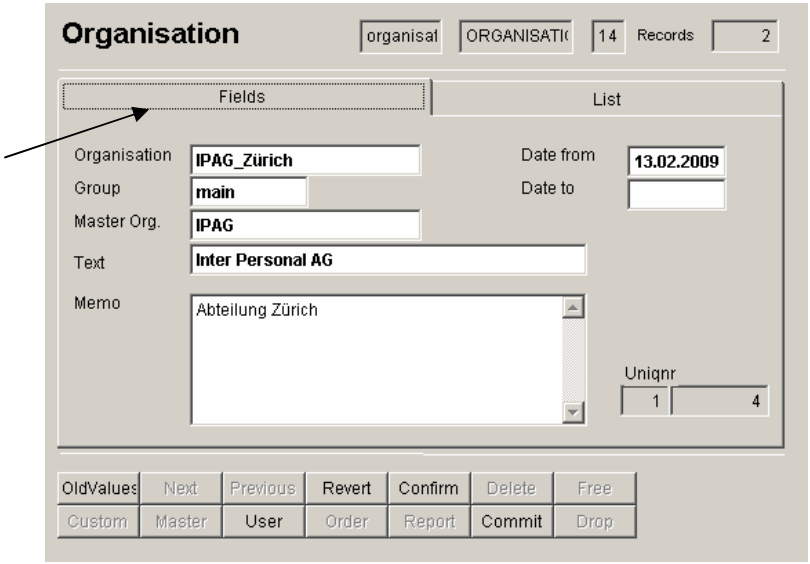


"Period form"

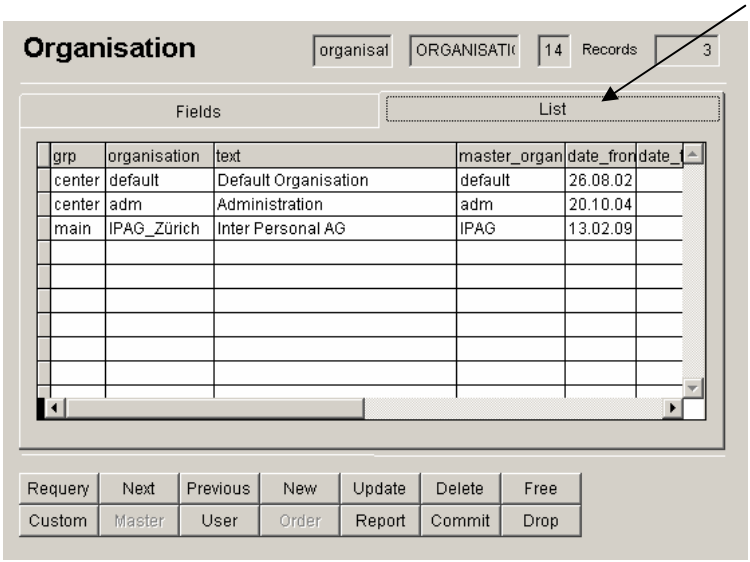
You can access the period form from the **selection tool**, setting the environment to **"tables"** and using a double click on the line **"period"**. Now click on the button **"new"** to enter a new period. Confirm with the **"update"** key. Referring to the **data model** the period can be considered as the **intermediate** between the **mandate** and the **batch**.

f. Organisation

Throughout the application, all bookings require an entry for the organisational unit. **Organisation** is mainly used for the representation of the **internal structure** of the (mandate) company, defining the **business unit**, **team** or department responsible for costs or revenues. Other terms used for organisation are **"cost centre"** or **"profit centre"**. The organisation can be presented in a **tree structure**, and **users** assigned to the **organisational units**.



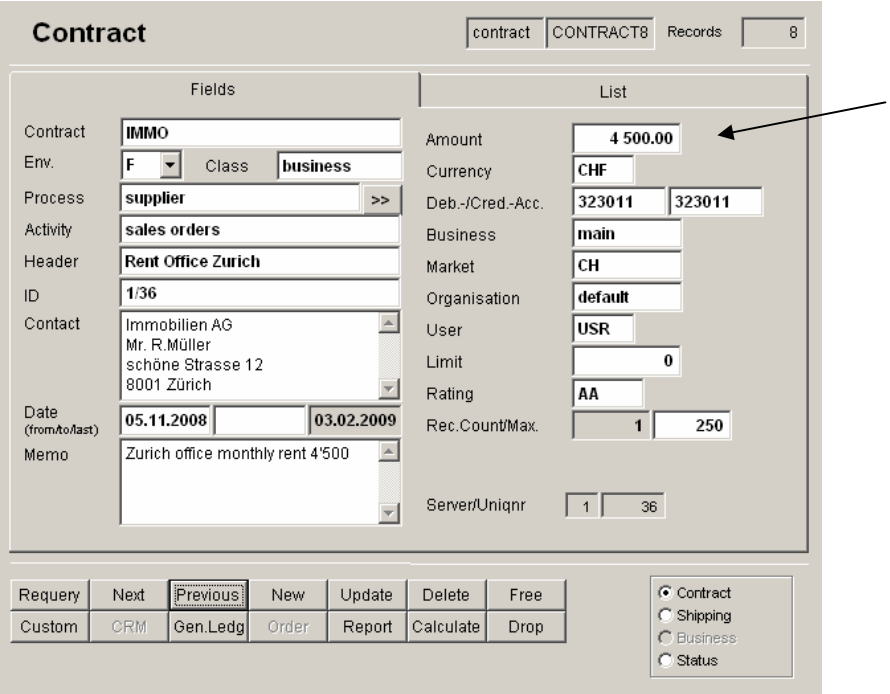
"Organisation" form with page "fields"



"Organisation" form with page "list"

g. Contract

"Contract" is used to for the representation of a single **business relation**. This might typically be the **customer** and the **vendor relationship**. In the **payroll process**, "contract" is used to represent the **employees**. Contracts are identified by a **unique string** (ex. Client.1234). Contracts can be used for the handling of **billing** as well as **shipping destinations**. At the contract level, a **default entry** for the **debit or credit account** and the **organisation** can be specified. During the entry of **general ledger bookings**, the system will **lookup** the account or organisation based on the **given contract value**. This will help **increase the productivity** of data entry and the **consistency** of the account and organisation usage.



Field structure of the contract
(with default rent amount for booking entries of CHF 4'500)

Together with the contract, information is stored about the **business performance** of the contract in the current period. This includes purchase or sales totals, accounts receivable and payable totals, and cash balance. While the contract only holds the information about the **current period**, the system traces the performance of **previous periods** in the **"status"** table.

Attached to the **"contract"** is the **Customer Relationship Management (CRM)**. The **CRM** combines the information of a **person, address and position** into a **contact record**. The contact record then is **linked** to the **contract**. While the **contact record** holds more **detail information**, only the **main address text** is copied to the **contract entry**. **Contact** can **import** data from **external sources** with help of the **clipboard** (in Switzerland: TWIX).

The module **Customer Relationship Management (CRM)** is not included in the **„Public Version"** of **sqlFinance**. To make it easier for a new user this version only uses the **contract record** with the **client's address** and a **text field** for notes, but does not require an entry in the contact, person and address tables.

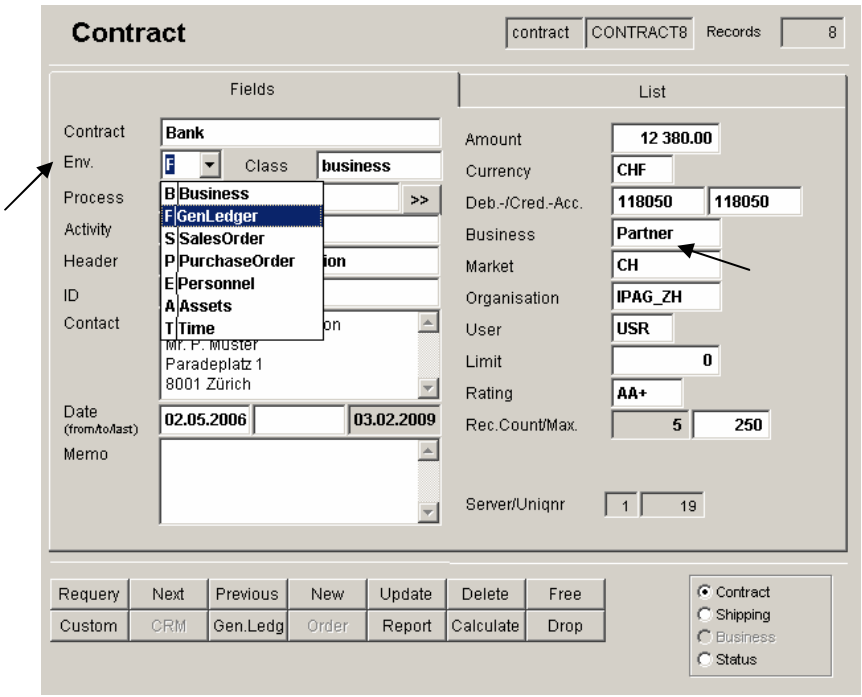
In the **„Full Version“ sqlFinance** uses **two types of contracts**, mainly in the modules **“purchase orders”** and **“sales orders”**, where the **batch form** has **two contract fields**, one for the **customer address** (“contract”) and another one for the **shipping address** (“shipping”). For the bookings entered, the **contract value of the batch** is used as **default**. Only the **general ledger** uses the **shipping contract** for the **payment function** (setting the bank account for receivable or payable transfers).

h. Business

Business is a **parameter** very similar to the **contract**. It will use the **same reference table** and provide **lookups** in the **contract table** as well. In the **contract form**, the **“environment”** value is set to **“B”** like **“business”**, whereas **contracts** are classified in the same field according to their **main application** like **“F”** for finance, **“S”** for sales, **“P”** for purchase, **“E”** for personnel/employees, **“A”** for assets and **“T”** for time.

The **“business”** is like the **“contract”** a detail to the **“contact”**. In many cases **contract partners** will be registered under the **“contract”** parameters and the **“business”** only used to structure company **areas of operation** like “consulting”, “partner”, “export” etc.

Under certain circumstances however it can be useful to **book purchase transaction directly to client projects**. In this case use the field **“business”** for the **client project** (or a group of projects). Without any necessity to pass over project or clearing accounts the **purchased goods** used in the **projects** can then be registered in the field **“contract”** and get directly booked to the **project**. In a similar way it is possible to use the field **“business”** for the establishment of a **profit centre analysis**. In many cases however it is easier to use the field **“organisation”** for this purpose.



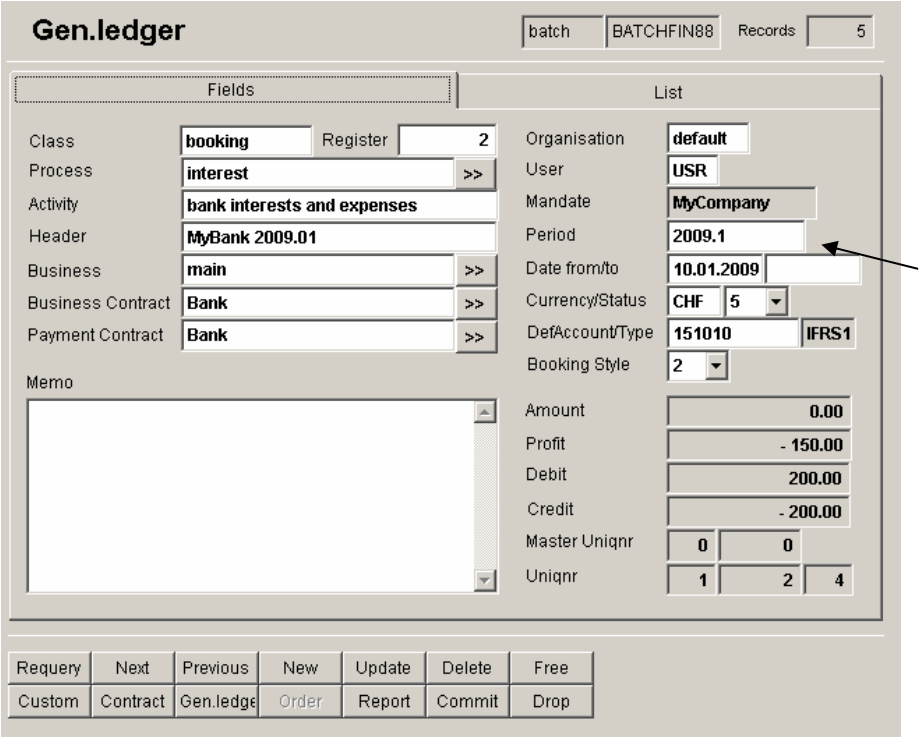
" Business" form

i. Batches and bookings

A number of **bookings** are hold in **batches**. Batches can be considered like **pages**, holding **headers** and a number of **lines**. In the **sqlFinance selection tool**, the **"environment"** listbox can be set to **"batches"** to get a list of available **batch views**. The **master/detail function** makes it possible to switch between the **"contract"**, the **"batch"** and the **"bookings"** form.

In many cases, the **text field** of the batch is used to **store longer text**, like letters, reports, notes, etc. The text is stored either in **ASCII** or in **HTML** form. A batch must be **attached** to a **period**. A double click on the **text field** will activate an **edit form** with **features** to support the editing process. **With copy/paste text can be moved to other applications or** imported into **sqlFinance** to get stored in the batch table.

Batches are always related to a **period** and a **process**. For **audits**, use the **report** based on the **period** to check through a number of **batches** or **batches grouped into processes**. The **text field** of the process could include **instructions for the bookkeeping**, the batch **comments of the accountant** for the understanding of the bookings. While the **"process"** describes a more **general booking pattern**, the **"activity"** describes the **current set** of bookings. The **header** is used as a **title** in a number of reports. As an example, for the receivables process "interest" use the activity "bank interest and expenses" and as a header "myBank 2009.01".



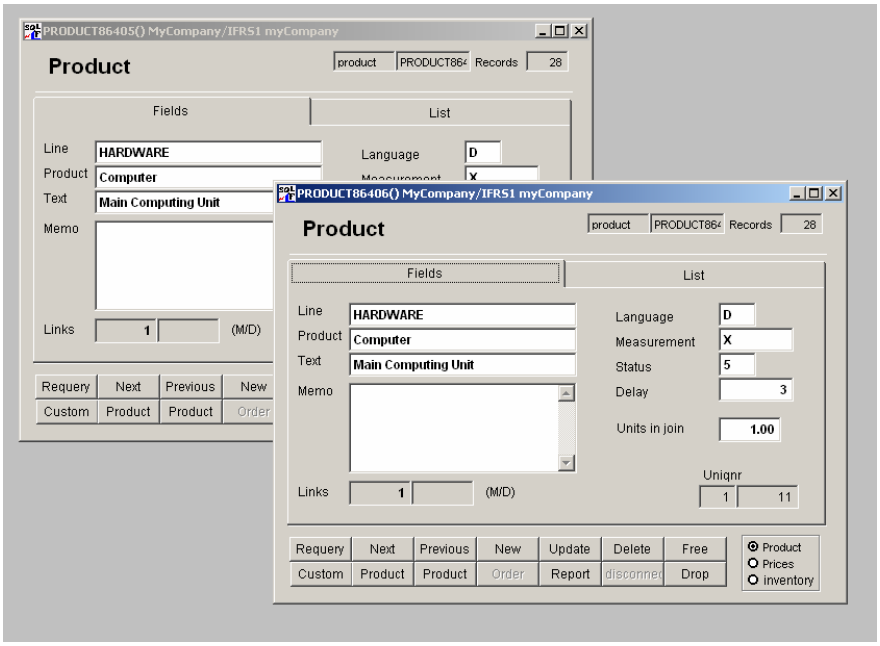
„General Ledger" batch form

Batch forms change in **appearance**, according to their **application** (general ledger, purchase, sales, receivables, payables, portfolio management, payroll). According to the **batch,** a specific **booking form** is selected. Some batches access **several booking views** (ex. a sales batch will access accounts receivable, tax and sales order). Use the **selection pad** on the right bottom of

the form (and double clicks) to get to the right booking form. Batches may **integrate** the bookings of **different views** to get to a **total of zero**. The total of zero is **tracked** on **batch and booking forms** (bottom right).

j. Logistics

Products are used in **sales** and **purchase** applications. With **drag and drop** operations products can be **aggregated** to a **bill of materials**. Dragging products over a booking will trigger the entry of a **new record**. In the **inventory table** stock may be **tracked**, as well as **mean purchase prices**. The **price table** gives information about price structure, depending on the **market**, with possible **discounts** per market or customer group. **Products** belong to **product lines**, and the **line table** holds information on which **accounts** product purchase or sales operations should be **registered** in the books. The **product line** also holds information about the **tax rate** applicable.



Products from the bill of materials, opened in a master/detail cascade (set with a PC)

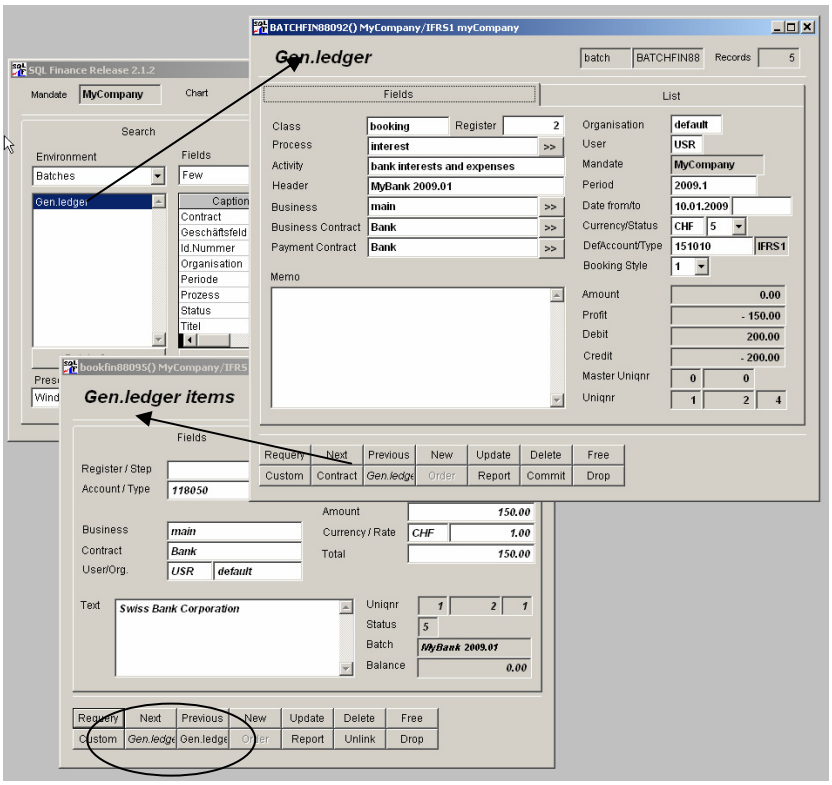
III. Application Modules

1. General Ledger

A new user might want to **start entering bookings as soon as possible**. For this purpose, the software is shipped with a **sample database** and the mandate **"myCompany"**. When the program is started, the user is **logged in automatically** and finds the **"selection"** tool presented on the screen. Two **sample batches** with bookings show him, **how data can be entered**. Use the **"selection"** tool to open the **batch form**. Set the **environment combo box** to **"batches"**, then double click on the **"sales"** position in the combo box below.

Double click on the **option group** at the right bottom (**"sales bookings")** or use the **detail key** (third from left in second line) to get the **booking records**. With the appearance of the booking form the batch form disappears. Use the **master/detail keys** (second and third from left in the second line of keys) to **toggle between the forms**. With the key **"free"** you can **disconnect** the forms and get them **both visible**.

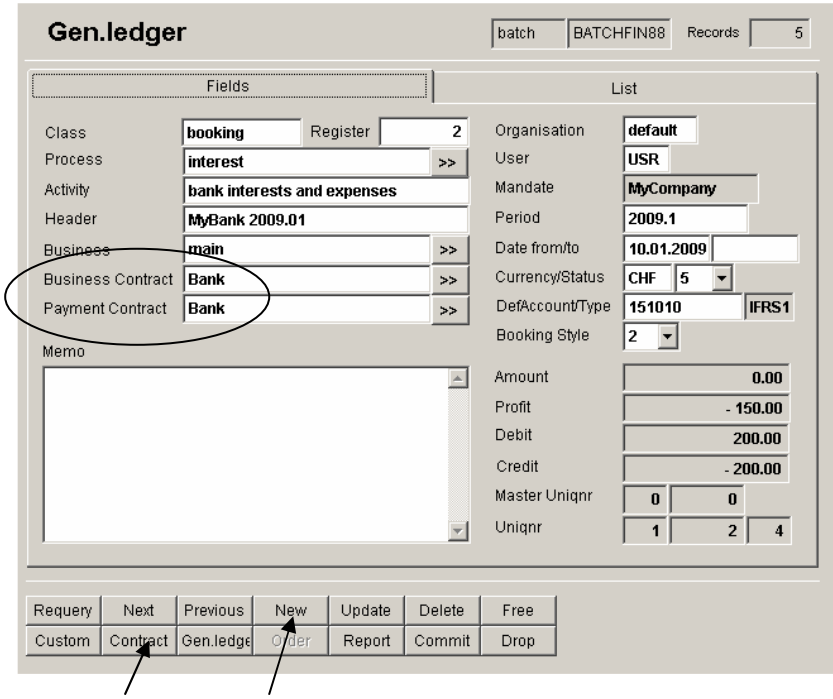
With help of the **"add"** key, create a new batch, then use the **"detail"** key (**"bookings"**) to open an empty booking window linked to this batch. Now enter the bookings. For the fields, **default values** like "2009.1" for the period, "default" for the organisation and contract, "USR" for the user will be accepted.



Step one: in the selection tool, set the environment to **"Batches"**, double click on the item **"Gen.Ledger"** to get the batch form **("Gen.ledger")**

Step two: select **"gen.ledger items"** as a detail of the **"gen.ledger"** batch (in the gl batch, click on the detail key)

From the **batch form**, the **contract** can be viewed using the **"master"** key. It is also possible to enter **search values** (optionally completed with one or more **wildcards**) into the **contract field** and use a **double click** on the field to get to the **contract form**. Save the data with the **"update"** key and return with the key **"close"**. The **booking form** will reappear, and the software will **transfer** the organisation and account default values from the contract. Please consider that with the use of a new chart of accounts the account defaults in the contract table will have to be changed as well.



General ledger batch with business and payment contracts

Now you can start with the entry of bookings. **Default values** from the batch form (e.g. "2009.1" for the period, "default" for organisation and contract, the contract's account value for the field account) will be set as soon as you click on the key **"new"**. Modify the record and confirm your data entry with the "**update**" key.

The **IFRS chart of accounts** can be used as it is or adjusted to user requirements. To get a **list of available accounts**, put the **wildcard** value **"%"** into the account field and then use a **double click**. Select an account record, and use the key **"release"** to get back to the booking window. The selected account value will be **copied**.

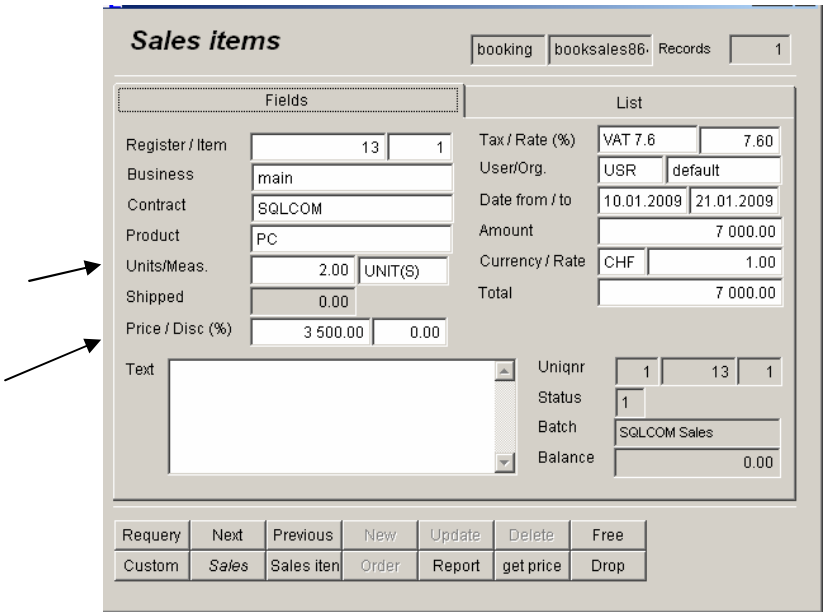
When a **new record** is entered, in order to get the **values** of the **previous record**, use the **"oldValues"** key (or **ctrl-q**). The **effect** of the **first records entered** on the booking form can be **tracked** right away on the **printed profit and loss statement**. Just get the **period form** with the correct period (e.g. with a double click on the period field of the batch), click on the key **"reports"**, and start the favourite report **"balance totals"**.

2. A simple sales transaction

The **"full version"** of **sqlFinance** includes **purchase and sales application modules**. Purchase and sales batches are entered similarly to general ledger batches. Always start from a **batch-booking cascade**. First open a **batch** with help of the selection tool, then select the **detail bookings** using the **'detail'** key. Proceed as follows for a simple sales transaction:

a. Get a **sales batch form**, enter a **new record**. The **default values** should be sufficient to allow an entry. Note that the **total field** of the new batch is **set to zero**.

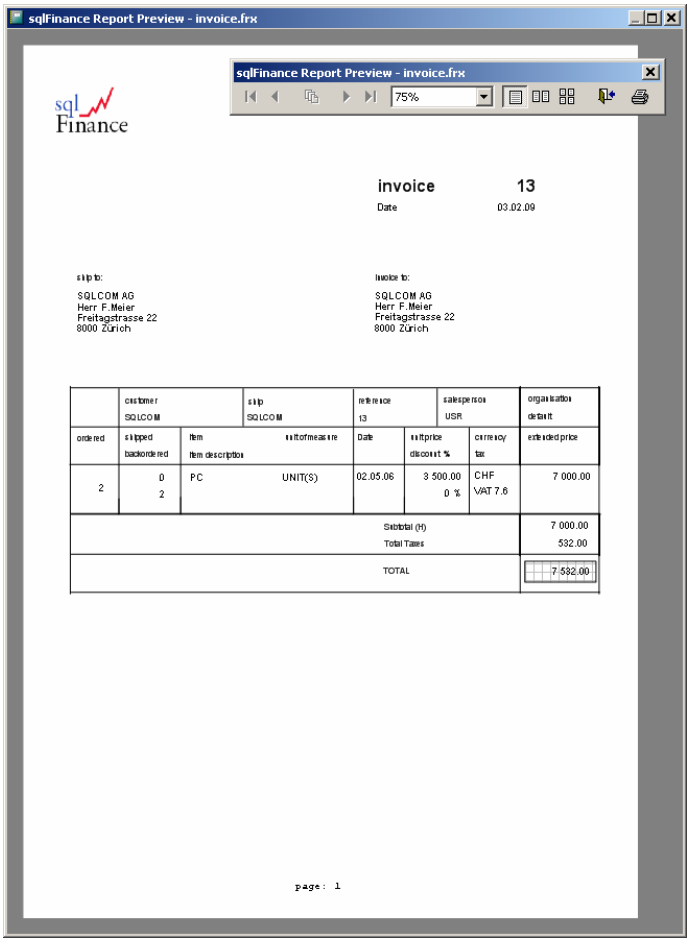
b. With the **detail/sales item key**, get to the **booking form**. Click **"add"** to get a **new entry**. For the **new record**, use the product **"PC"** and a **price** of CHF 3'500.-- (as an example). In the **"units"** field enter the **quantity** of PC's to sell (2), in the **discount** field the sales discount for the customer.



Entry of a sales booking: 2 PC's at a price of CHF 3'500

c. After the entry is completed, note that the **new batch total** (bottom right) includes 2 x CHF 3'500 + 532 (VAT 7.5%). The **total** is **CHF -7'532** (a **negative value**), since this is a **sales (revenues) entry** in the chart of accounts (**type 4**). The **account** is **set automatically** according to information in the **product line** (line is a **master** of the **product table**). To modify the account settings, select the **"line"** table for the **product line** "HW" (hardware), or **drill down** with **double clicks** on the **"product"** and then again on the **"line"** field.

d. With use of the reports tool, an invoice to the customer is printed. The invoice can include billing and shipping addresses, and information about backordered shippings. Multiple taxes are possible per bill, and foreign currency calculations. The product information is completed with units and measurements, as well as discounts per products or markets or clients. The sales order is available in the "full version" of sqlFinance and includes a full CRM (Customer Relationship Management), BOM (bill of Material) and a price expert.



Report with invoice (from the sales orders module)

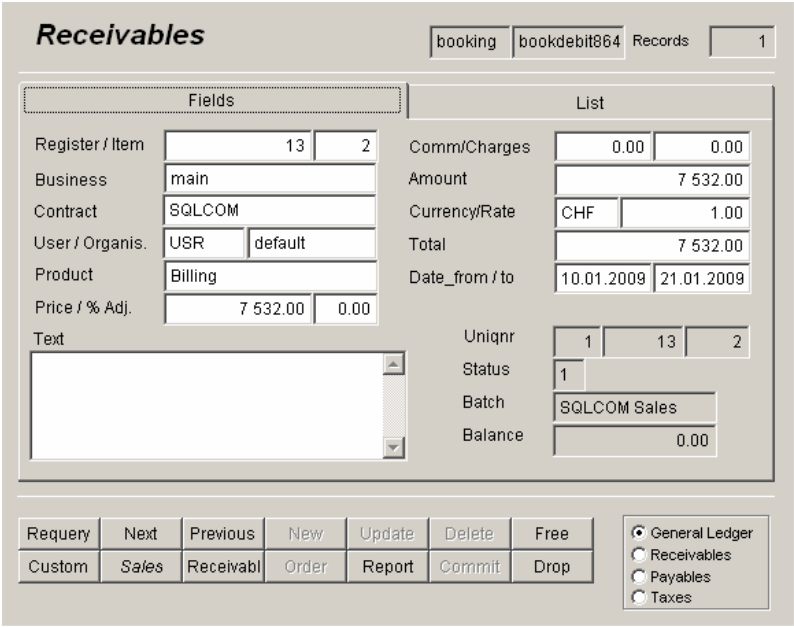
e. Release the sales booking form, and get - from the sales batch form - a **receivables form** (through a click on the "receivables" position in the list view at the bottom right of the form). The options of the list view include:

- Sales items
- Receivables
- Taxes
- Picking
- Shipping

Picking and shipping are not supported in the PC version.

f. Enter a **receivables** record. For the product choose **"bill"**, and enter the **total** of CHF 7'532. You may **drag the amount** from the **balance total** (-7'532) to the **total field**. Note that the **amount** is **changed automatically** from -7'532 to its **counter total** of 7'532. Use the **right mouse key** for **drag and drop**, use **double clicks** on the **money fields** to change from a **(-)** to a **(+)** sign.

g. After entry of one or more the receivables, the **batch total** is **zero** again. Now the batch can be **committed** to the **general ledger**. There are **two ways to commit**: either, the **bookings are sent unchanged** and as part of the **current batch**, or only the **totals of sales items** (combined with all details of receivables and tax entries) are sent to a **new general ledger batch**. This batch will be **generated automatically** and attributed to the **reporting (master) period** of the **sales batch period**. Set the **status** to **1** and use the **update** key in order to commit the sales batch as it is. Use the **commit key** to start the creation of **general ledger bookings** with **sales item totals**. With **"recall"** in both cases the transaction can be **undone**.



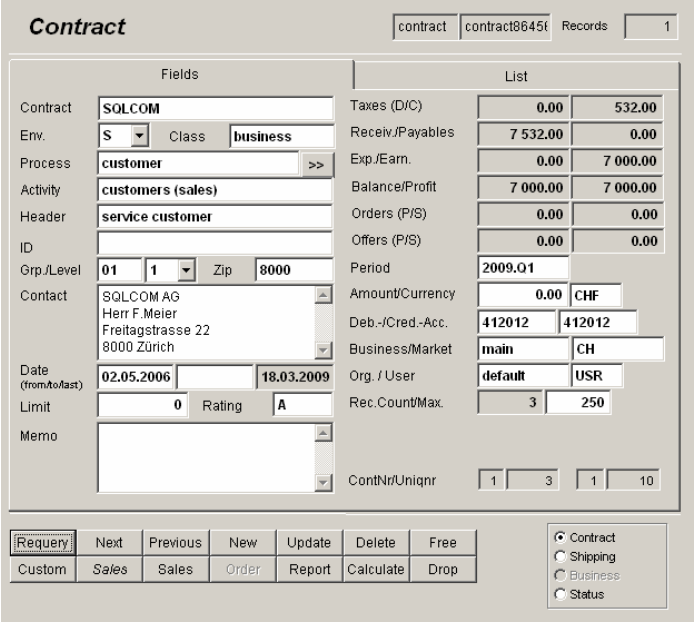
Receivables form with total amount of CHF 2'532.--

h. At the reception of a **payment**, drag the **receivables position** (CHF +7'532) over an **empty receivables** or **general ledger booking form** (normally connected to the new payments batch). Get an **automatic clearing** entry of CHF -7'532. Note that the **new entry** appears in **bold characters**, indicating the **connection to the receivable entry**. It is possible to **select** the entry in **another form** with use of the **"detail"** key.

i. Drag the **receivable** or **clearing entry** over the **"clearing"** tool to **check all links**. Now **clear the new batch** from CHF -7'532 **to zero** with an **entry of the payment**, that is a **receivable** or **general ledger booking** of CHF +7'532 (usually on the **bank account**). After a click on the **"add"** key, use the **"payment"** key to get automatically the right amount and a link.

j. Check the **contract table** to see how **sales entries** and **commit/recall commands** influence the **totals** of the **contract reporting**. **Before "commit"** only the order total will amount to CHF 7'000. **After "commit"**

the earnings field will amount to CHF 7'000.--, the tax field to CHF 532.--, and the (open) receivables field to CHF 7'532.--.



Contract form with statistics from the sample sales operation

k. The **period** (2009.1) may be committed with the **"commit"** key from the **"period"** window. Use a **double click** on the **period field** of the **"batch"** form to get to the **"period"** form. Note that a period can have an **intermediate closing** (ex. monthly sub periods 01 - 12 to yearly period 1, or a **year closing** (yearly period 1 closing to new yearly period 2). Intermediate closings include totals for assets, liabilities, expenses and earnings, year closing include totals only for assets and liabilities (and a profit total).

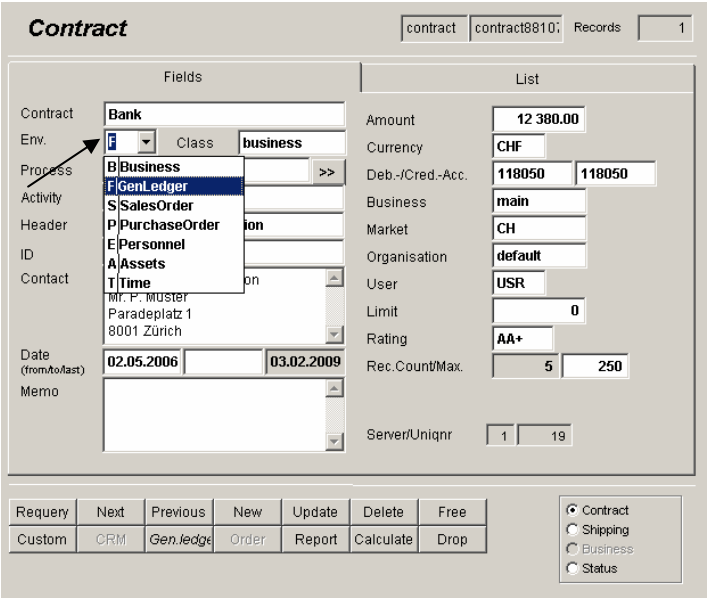
3. Debit/Credit Clearing

In **all booking forms** it is possible to **connect entries** with a **link**. A link makes it possible, to **retrieve** later one or more booking records **connected** to a another booking entry. If a booking entry is **linked**, it will appear in **italic characters**. As soon as the records are **unlinked**, they will appear in **normal characters** again. Every time a booking is **dragged** over **another booking**, the application **asks the user**, if he wants to **enter** a **new entry with a link**, a **new entry without a link**, or just to **link the two records**.

The **linking of bookings** is important for the **clearing of debit positions** (sometimes credit position are tracked this way as well). When an **invoice** is sent to a customer, a **debit position** is kept **open until payment is received**. If the payment is **not corresponding** to the **exact due amount**, an **adjustment** entry may necessary. The following instructions show **how to clear open debit positions**:

a. Get a list of open **debit entries** (in the selection tool, choose bookings: accounts receivable).

b. If not entered yet, create a contract for your **bank account**. As a **detail to the bank contract**, create a **general ledger batch** and open the (detail) **receivables form**.



Contract used in gen.ledger batch (with environment set to "F")

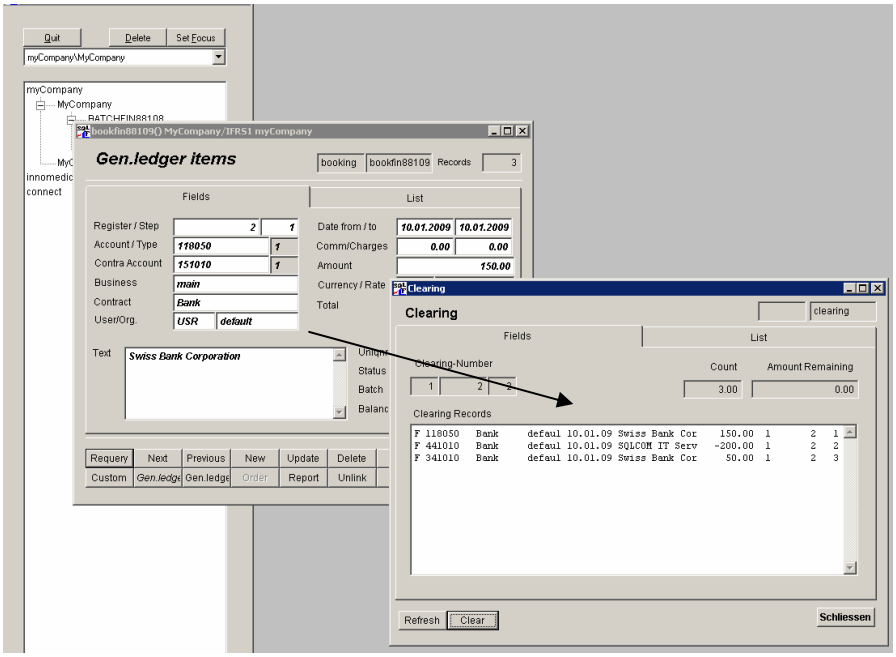
c. **Drag** the **debit record** over the **new receivables** form: a **payment entry** will be prepared. The new record is **linked** to the debit record, the **total** will be the **negative amount** of the **other record**. Confirm the **"add"** operation. The **contract value** of the **debit record** corresponds to the **client contract**. Instead of drag and drop but with the same result, in the **receivables** or even **general ledger form** with the selected debit position, the **'add'** key can be used to get a new entry, and then the **'payment'** key to get the new entry **linked** to the debit position. In both cases, change the

contract value to the **bank contract**. If the bank contract is entered in the **batch** as a **default for the shipping contract**, you may **load** it with the shortkey **ctrl-F10**. As soon as the two records are **linked**, they appear in **italic characters**. Note that you have to use the **"requery"** key in the **source record** to see the effect.

d. To **erase the link**, use the key **„unlink"**. Unlinked bookings appear in **normal characters** again. To get from one form to another, use the **"master"** and **"detail"** key or the shortkey **ctrl-F2/F3**.

e. In case that a client only pays a part of a bill, **either modify the bill** (with automatic adjustments of tax liabilities) or **clear the difference to a special account** (ex. "sales loss, taxed 7.6%") and use that total later as an adjustment in your tax declaration.

f. Before the batch is **closed** the software checks, if the **link totals** in the batch all result to an amount of **zero**. If the batch includes records with a **non zero total over the linked bookings**, a **warning** is sent to the messenger. If necessary, a link can be erased with the "unlink" key.



Booking form dragged over the "Clearing" tool

g. You may **drag linked bookings** over the **clearing tool** (activated from the menu **"view"/"clearing"**) to get an **overview** over the **linked bookings** and their **total**. As soon as a booking is dragged over the **Clearing tool**, a **list of all linked bookings** is produced. With reference of the **server/uniqnr/trans value** these bookings can be **retrieved** through a **query** from the **selection tool**.

h. In a similar way, the **clearing form** can be **dragged over a booking form** to **link the target booking** to the **list of already linked bookings**. The **link** entered in the **target booking** has to be **confirmed** with the **"update"** key.

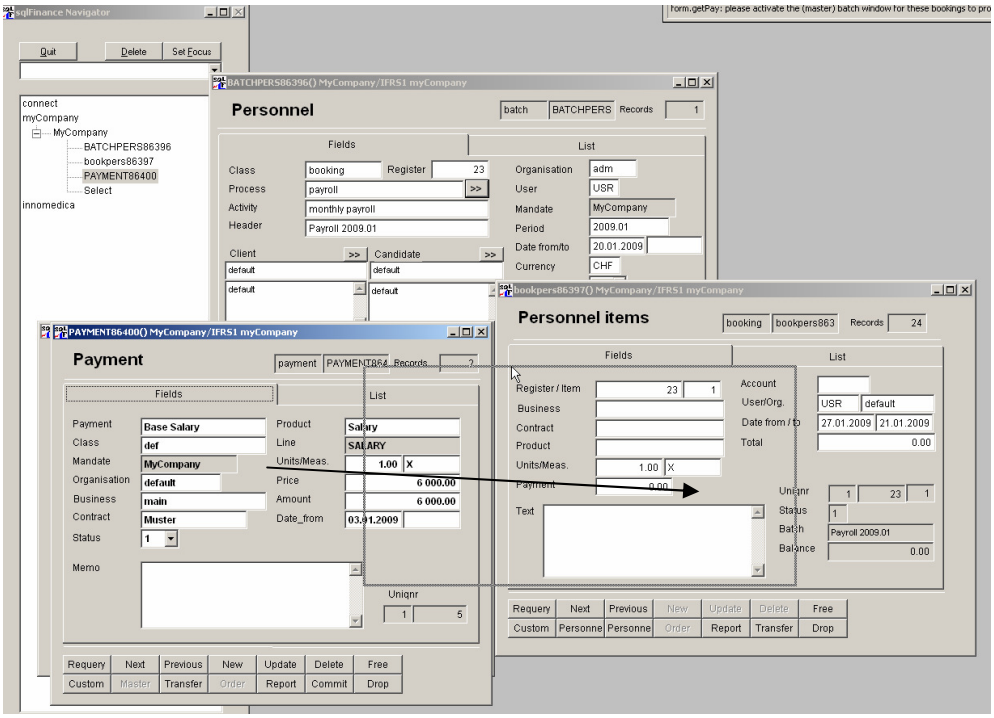
i. **Receivables** and **customer payments** for **key accounts** are most easily registered with **one batch per customer**. If there is are **very many customers**, a **batch per month or per area** can be used as well. For each batch, a **list of open and cleared bookings** can be generated with use of the **report tool** (key "reports" in the batch form). This is useful for the **payment control**. At the end of the year, the batch can be **dragged** over another batch to get a **batch entry in the new year** and a **closing and reopening booking**.

   An **interactive form** asks, if this function should be run for the **current batch** only or **all batches in the list**. With entries in **check boxes** the user can indicate, if **only batches** should be copied or **batches and bookings** as well, or if **instead of copying all bookings just one closing and reopening booking** should be **generated** (see the chapter "Drag and Drop" later in this manual).

4. A short Introduction to Payroll

sqlFinance Payroll uses the **contract** structure to describe **employees**. If necessary, more than one contact per **employee** is allowed. Monthly or yearly **payments** are **associated with the contracts**. Payments include both a **product** and a **product line** value. Depending on the **product line** value **transfers** can be selected and bookings **generated automatically,** using the product line **account assignment**. **Transfers** are used to **define social security and insurance charges deducted from monthly salary payments**.

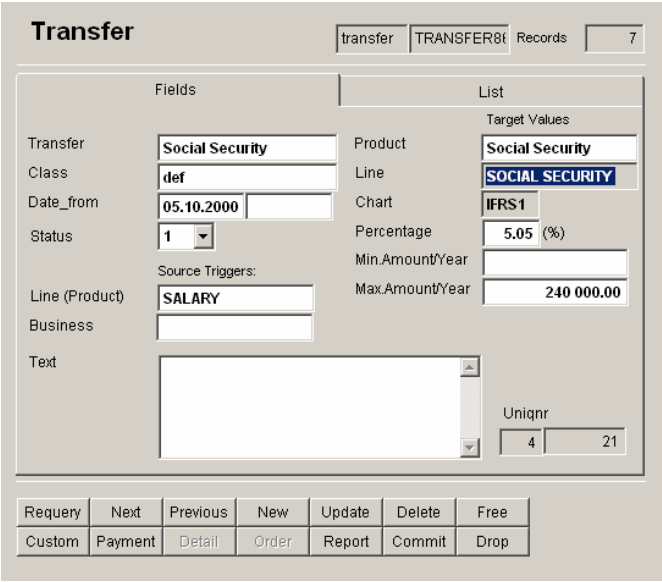
To run a payroll batch, payment records are dragged over an empty payroll booking form. After copying, with a click on the **"transfer"** key, the **automatic insertion of transfer records** can be started. From the **payroll batch record**, **reports for employees and lists with totals** can be printed. If **errors** occur, with a delete attempt of the batch all the entered bookings can be erased and the **procedure restarted**. At the end, the batch is **committed** and the **sub period** of the payroll **closed to the main yearly period**.



Form "Personnel items" as a detail of the "personnel" batch.
Drag "payments" over the empty "personnel items"
to generate payroll bookings
(note the drag grey frame)

a. Basic payroll proceedings:

- Enter and manage **payments** in the **payment form**.
- Select a **subset of due payments** (ex. status = "1").
- Select a **personnel batch** (normally a new batch with no personnel booking entries yet). Use a **new sub period** for this batch (ex. 2009.01).
- **Drag payments over the personnel booking form**.
- **Reselect the bookings** from the batch.
- Click on the **'transfer'** key to get transfer entries **automatically generated** for each payment (social benefits, insurances etc.).
- Answer with **'yes'** or type **ctrl-w** at the **browse windows** presented for checks during the transfer process.
- **Reselect** again to see the **new payment and transfer entries** in the personnel booking form.
- Get back to the **personnel batch form** and print the **payroll** sheets or lists (click on the **"reports"** key to get to the **report writer tool**).
- When an **error** occurred, click on the **"delete"** key of the batch: the system will then propose to **delete all the corresponding bookings** as well.
- When no errors occur, **commit the batch** with the **'commit'** key.
- **Close the sub period** to the **main yearly period**.



Transfer record for Social Security charges
(trigger all payments of the product line „SALARY"; enter transfer records with accounts assigned according to the product „Social Security" and its product line „SOCIAL SECURITY")

b. Payroll period management

- A **monthly payroll** is usually stored in a **separate and intermediate period** (ex. monthly period 2009.01, closing to 2009.1, the annual period).

- After the monthly payroll, close the **monthly period** to the **yearly period** and get the summary entries into your main accounting. Closing parameters in the period form can be set to **closing totals per organisation** or **closing totals without organisation**.

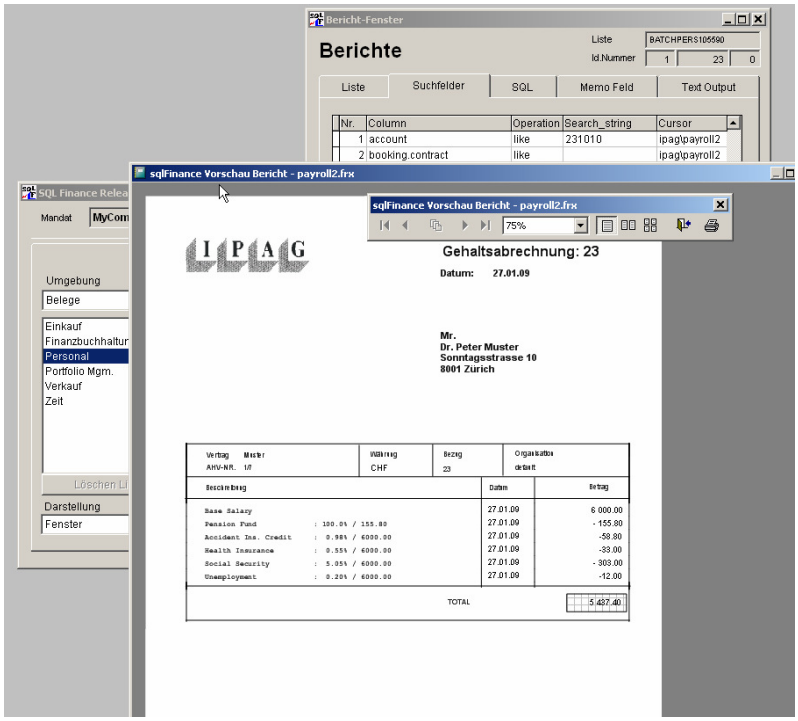
- The **intermediate periods** of the payroll should **not be used for other bookings**. Rather use **several intermediate periods** per month to keep your data **well organised** (e.g. 2009.01a, 2009.01b).

c. Payroll printing

- Print the **monthly payroll** with the **payroll report** in the **personnel batch form**.

- Print the **yearly payroll** from the **period form** with the report **"Payroll Period"**. Be sure to start this process from the **right period record**, usually the **yearly period** all the monthly sub periods will report to (through the 'close period' process).

- There is a special report **"payroll official report new"** for the new **Swiss payroll reporting**, filling in correctly the fields of the official payroll form.



Example of a payroll report selected from the batch record (with search value for account set to "231010")

d. Payroll data structure

- Use the **'contract'** field to handle **employee contracts**. The contracts can be grouped to **business fields** ('business') and **organisational units** or **cost centres** ('organisation').

- Payment entries include **product and line description** (ex. fix salary per moth: product 'Salary', product line 'SALARY'). In the **'line'** table, enter the **debit and credit accounts** that will be **used later** for the **creation of payroll bookings**. Use the same approach for the **'transfer'** table.

- **Payment and transfer records** are **joined** based on the **trigger line entry** of the **transfer form**. The **trigger line of the transfer** corresponds to the **line of the payment product**. In other words, **transfer records** are selected and joined with **payment records** according to their **trigger line value**. Make sure to enter **no value** for the **trigger business field** unless you want to **limit transfers to the payment business value**.

- The **new bookings** generated by the **transfer record** will get account values **not from the trigger line value**, but the **target product and line values of the transfer record**.

payment record		transfer record
payment product		
payment line	→→→	transfer trigger line
		transfer product
		transfer line

- Use **three types of accounts**:

account	type	line field
- salaries	(expenses)	debit
- payroll employees	(liabilities)	credit
- social benefits	(liabilities)	inventory

- How to **proceed with the bookings**:

salary payments:	+ salaries	- payroll empl.
social benefits (employee part):	+ payroll employees	- social benefits
social benefits (company part):	+ personnel expenses	- social benefits

Social benefits (employee part) include the employees participation in the payment of the benefits (usually 50%), the social benefits (company part) the other half (usually paid by the employer).

The **accounts** should be registered in the **product line**. Products should be entered for the **salary payments** and the social benefit **transfers**. In the **line form**, there are **three fields for the accounts**:

a. 'debit': personnel expenses
b. 'credit': payroll employees
c. 'inventory': social insurances clearing

- Please consider, that the **salary** appears as a **positive amount** in the **expenses**, but as a **negative amount** in the **payroll clearing account**. The **social benefits** charged to the employee, on the other side, will appear as a **negative amount** in the **social benefits**

clearing, but a **positive amount** in the **payroll clearing**. In the **employees payroll report**, this is adjusted of course to **inverse signs**, presenting the **salary as a positive amount** and the **charges as negative amounts**.

- Consequently, the **payment of the net salary** is a **charge to the bank account** and **a positive booking** (for one or a number of employees) in the **payroll clearing account**. This will bring the **total of the payroll clearing account** to **zero**.

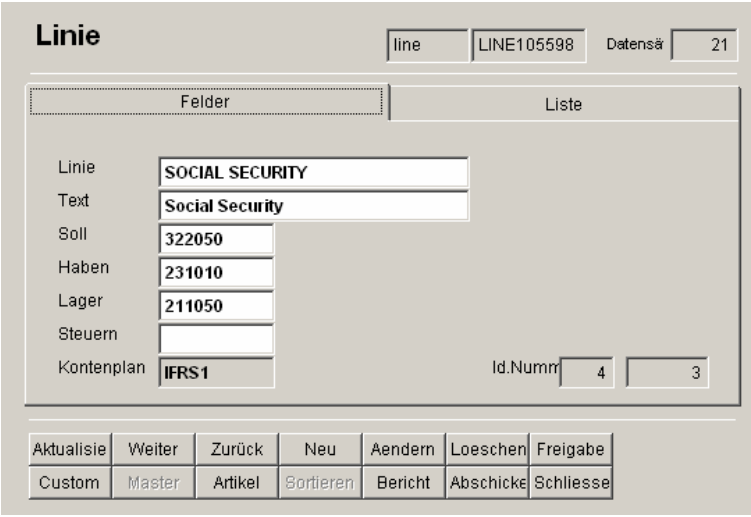
- **Example (social security):**

  3/322010: personnel expenses
  2/231010: payroll employees (liabilities)
  2/211050: social insurances (liabilities)

  Based on this line entry, **two payments** will be generated:

  a. debit to inventory
     (company part, not appearing in the payroll statement)
  b. credit to inventory
     (employee part, appearing in the payroll statement,
     based on account 231010)

  Records in the **„line"** table **accept null values** for accounts. Null values will be **skipped** in the transaction. Debit to inventory will charge the social benefit to the employer, credit to inventory to the employee.



Example of a social benefits implementation for a Swiss payroll

e. Example of IFRS-Accounting for Payroll (in the "myCompany" database)

Debit Accounts:

322010	salaries
322050	social security
322060	health insurance
322070	accident insurance
322080	pension plan
322090	other personnel insurances

Credit Accounts:

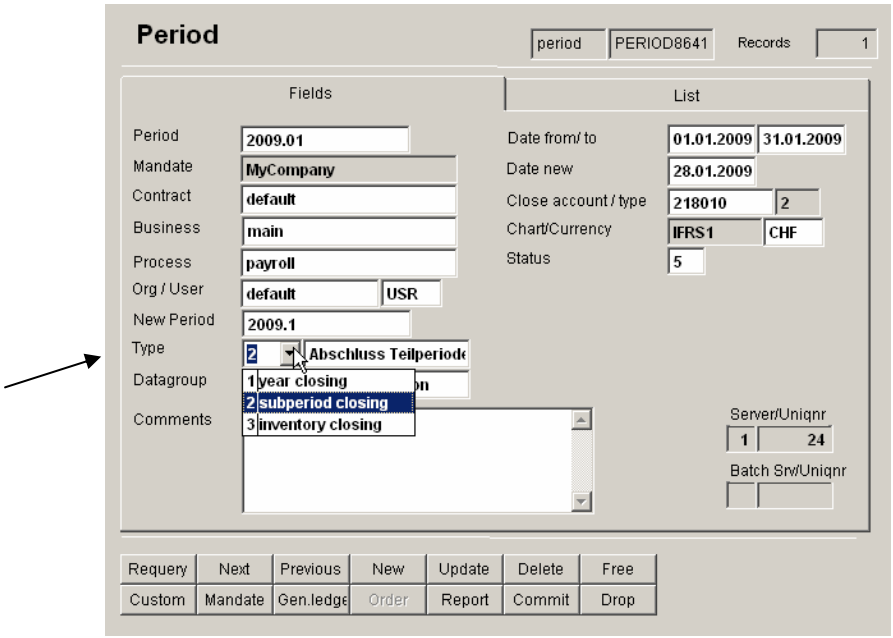
231010	payroll employees (clearing account)

Inventory Accounts:

211050	social security
211060	health insurance
211070	accident insurance
211080	pension plan
211090	other personnel insurances

Liabilities of the inventory accounts (211050 to 211090) show how much there is to **pay to social insurances**. Another **liabilities account** (ex. 211150 to 211190) may be used to **list the payments already made** and compare the two groups to check if the payment estimation and the payments made correspond well.

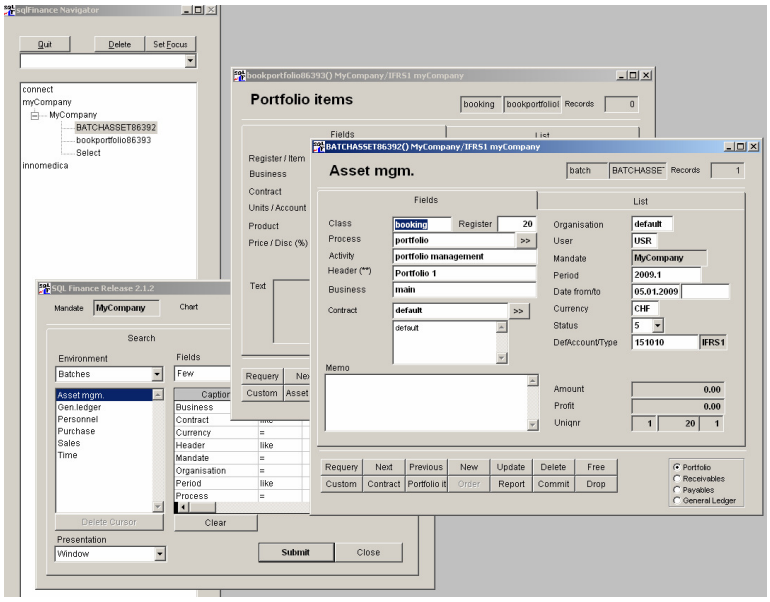
The **monthly payment to employees clears the payroll account** (231010) to **zero.** The **payroll listing** for the employees is – as a matter of fact – **a simple list of this account**. Consequently the **monthly total of this account** indicates the **total payment due to the employees**.



Example of an open subperiod in the payroll application (with closing per sub period and organisation)

5. Asset Management

sqlFinance includes, in the full version, a module for **asset management**. Part of the module is an **interface** for the **import of trading information**. Like all other modules, it supports **foreign currency** and includes fields for **banking commissions, taxes or other charges**. Based on the **period**, **reports** for the **clearing of foreign currencies** to the **reporting currency** can be generated. The reports also make possible the analysis of **performance per title** and the separation of **asset performance** from **currency performance**.



Asset management cascade with batch and portfolio bookings

For the **portfolio management** a **booking example** over three years is joined to the **"public version"**. This example has been **kindly provided** by the finance company **"Innomedica Holding"** for education and marketing purposes. The example illustrates the following **business cases**:

- Trading of listed stock (health sector)
- Trading of own shares
- Issuing of convertible bonds
- Clearing of expenses for portfolio management
- accrual of currency gains and capital gains from stock trading

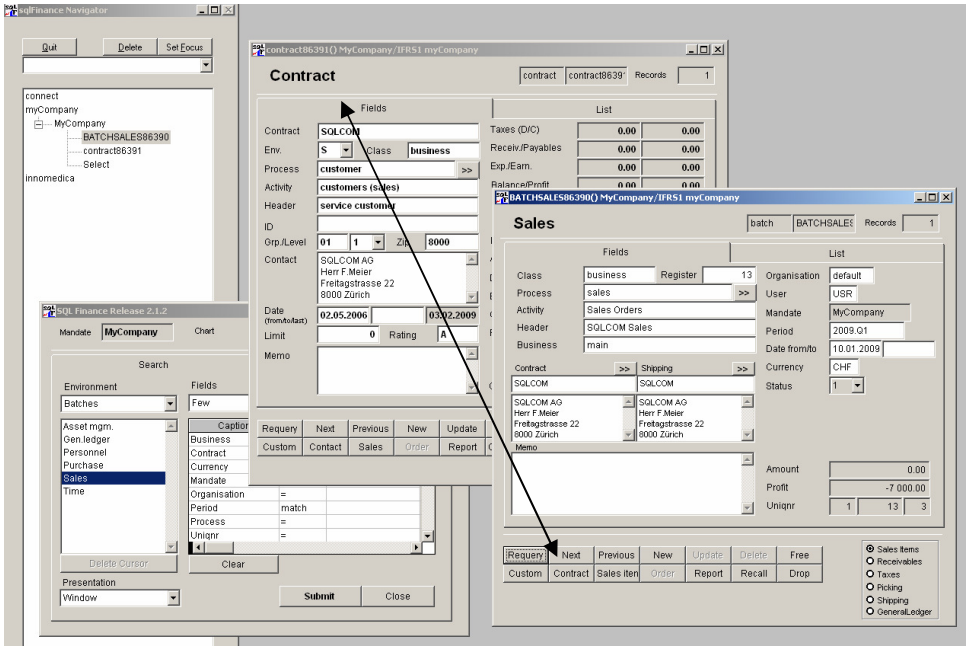
This example is practical for the study of **batches and bookings** and to get acquainted to the use of **print programs**. The sample data are accessible through the connection **"Innomedica"** in the connection tool. Then select the mandate **"Innomedica"**, and check the **three periods** 2003.1, 2004.1 and 2005.1. From the period form a **balance and income statement** can be generated using a report like e.g. **"Balance Totals"**.

IV. Key Functions

1. Forms and Cascades

sqlFinance forms are used to **present selected data** and have a **standard appearance**. They all include a pageframe with a **first page for the fields of the current record** and a **second page with a list of all records in a grid**. Every form has the **same keys** for update operations or the activation of other forms. A number of connected forms, selected with the master or detail key, appear in a **cascade**. **Only the current form is visible**. The forms are usually connected through a **master-detail relationship**.

The **drill down** from a master to its details can be made over **several levels** and is limited only by logical aspects of the **data model**. Master-detail relationships follow the main structure: **address – contact – contract - batch – booking**. Check the data model for a full understanding of master-detail relationships.



Simple cascade contract – sales batch with detail key at master form and master key at detail form

In the **cascade**, only the **current form** is visible. With use of the **"free"** key forms can be **separated from the cascade** and then get **used again** for the creation of **new cascades**. This may be practical in many situations, for example after a **double click** on the accounts field of a booking form, when the user wants to **keep the double clicked account record** (or the account list when a wildcard was used for the query) as a reminder.

As an example – get the **'batch' details** of a **'contract'**: first **select the contract** with use of the **selection tool**, then click on the **detail key** (third in the second line of keys) of the contract form to get the **detail 'batch' records**.

sqlFinance will select **all batches related to this contract** (checking the contract field of the batch against the contract field of the contract).

Every time a **new batch** is entered, the new record will **inherit the contract value of the master (contract) record**. Note that the **environment** value of the contract must be set to **S/Sales** in order to get **a sales batch** as a detail. Other detail batches can be selected using the **option list** on the right bottom of the contract form.

a. Standard keys of the form:

upper row:

- Requery (selects recent data from the disk)
- Previous (get the previous record)
- Next (get the next record)
- Add (get a new record to fill out)
- Update (store updated fields to the disk)
- Delete (destroy current entry)
- Free (disconnect

lower row:

- Custom (modify values for fields in the master database)
- Master (select the master table)
- Detail (select the detail table)
- Order (modify the ordering of the records
- Report (access the report window with reports for this form)
- Commit (start form-specific action like closing batches)
- Drop (close window, drop cursor)

b. Some **key action** can **vary according to the form**. The most important features are:

- **Batch and period forms**: commit open batches or periods to closed batches or periods (or back).
- **Personnel bookings**: create transfer bookings for personnel payments, as specified in the transfer table (see section about payroll).
- **Contact form**: The clipboard key will get Twix data (clipped with ctrl-c) and produce entries in the fields; an efficient feature to copy private or company addresses from any telephone book sources into the contact form.

c. The **contact record** needs, not like any other record, connections to **three master records** to be valid: **address, person, position**. They all can be set by **dragging** the **"fields"** page of an address, a person or a position **over the contact form**. Only when **all three connections have been set** (the address may have been inherited through a master/detail connection of the form), **a new record can be entered**.

d. Address, person, position and contact data is part of the **Customer Relationship management (CRM)** and not included in the **"free version"** of **sqlFinance**. The **"free version"** simply uses a **contract record**, including a **contract name** and an **address field**. No link to a contact or its

master address/person/position tables is necessary for the contract record to work properly. This makes it easier to get first familiar to the application.

e. **Shortkeys:** For most **key buttons on the forms** there is a **shortkey** available. For example, entering **ctrl-s ("select")** will start a query in the selection tool or in the report. **Ctrl-a ("add")** will add a new record on a form, **ctrl-u ("update")** will confirm the entry to the database. A complete **list of shortkeys** can be found at the end of the manual. The **shortkey** is indicated on the **messenger line** when the user clicks the **right mouse key** on a **key button**.

f. If master or detail forms are **connected**, the corresponding master/detail keys appear in **Italic writing** and indicate the **name of the master or detail cursor**. A **click on the key will switch control in the cascade to the master or detail form**.

g. In a cascade, using the **master and detail keys** of the forms it is possible to **move quickly up and down**. Instead of clicking on master and detail keys, the simple keystrokes **alt-F2 (master)** and **alt-F3 (detail)** may be used.

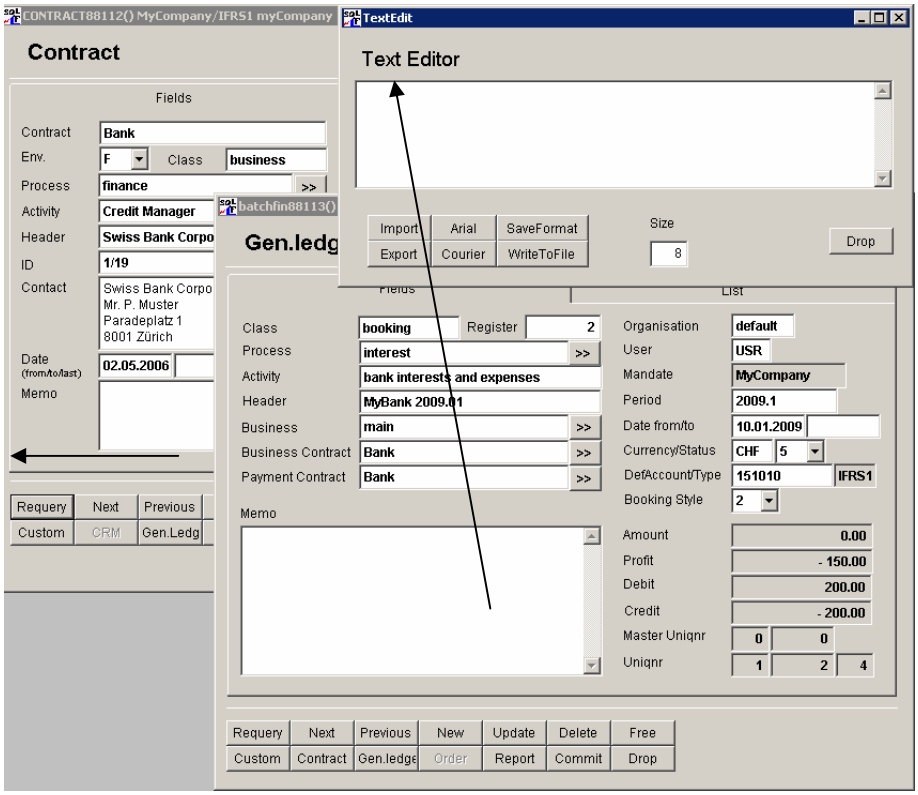
2. Fields

Fields appear mostly on **page one** of the form's **pageframe**. Fields belong to **field types**, used for the presentation of **specific values** like numbers, characters, dates etc. Field types have a **standard behaviour** in the way they react to user action like double click or drag and drop.

a. Text fields

Text fields often are referenced to **master tables**. For example, a **contract name** entered in a contract field of a booking record **references to a contract entry in the contract table**. Through a **double click** in the **contract field** (of the booking form) the user can **access directly the contract table** (opened as a detail**) with the selected record**.

The system will **accept the use of wildcards** in combination with the field value (here the contract name). Entering **"myCompany"** will have the same effect as entering **"*myCompany*"** or **"%myCompany%"** and bring **all contract records with a contract name including the substring "myCompany"**. When the contract form is **closed**, the **contract name** of the selected record is **copied to the initially double clicked field of the batch form**.



Text editor and contract lookup form started with a double click in the memo field of the batch form

b. Numeric fields

- **Integer numbers** are stored in **integer fields**; **decimal numbers** are stored in **money fields**. Only money fields offer special capabilities.

- **Double clicks** on money fields will **invert the value** (from positive to negative or vice versa).

- Many money fields can be **dragged** over each other. Drag the field with help of the **right mouse key** (the left key will start Microsoft standard editing functions) and place it over the target field. The dragged value is entered in the new field **as soon as the mouse key is released**. In the **booking form** this may lead to **adjusted calculations** in other fields (if for example you drag a new amount over the price field). The most common application is to drag the **running total** of a batch over **a final booking entry** to **get the total of all bookings for that batch to zero**.

- **Money fields** in the booking form are programmed for consistent calculation **(units * price = amount, amount * currency-rate = total)**. It is better to drag a source over the price or amount field than the total field, since **changes in the total will lead to the adjustment of the rate rather than the amount or price fields**. The rate adjustment may be practical in **portfolio management**, but consider that rate adjustments may lead to **undesirable rounding effects**.

c. Date fields

- **Date fields** always include **date and time values**. If time is not considered important, it can be omitted.

- **Double clicks** on the **date field** first delete the **date value**, then enter the **current date** (on the second double click).

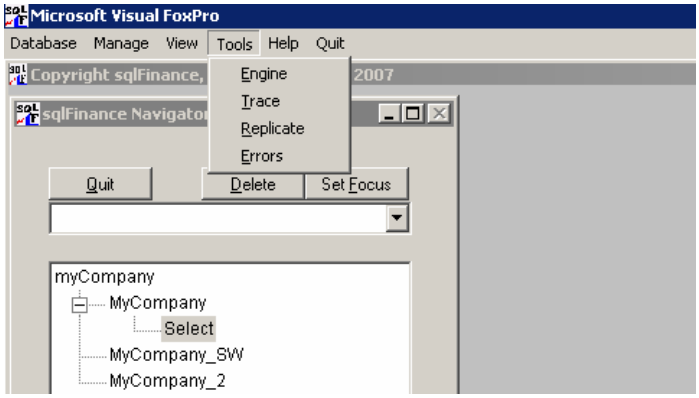
d. Memo fields

- **Memo fields** are used to store **multi line text elements**. A **client address** used in a letter is an example of a **multi line text**. Text stored in **memo fields** may be up to **many pages long**, depending only on the **capacities of your system**. However, no line longer than **8192 characters** is accepted.

  Double clicks on the **memo field** activate in a first step the **sqlFinance editor**, another click on the sqlFinance editor activates the **"notepad" shareware editor**. Note that a version of the **"notepad"** shareware editor must be **downloaded** and installed in the **sqlFinance** root directory to get this feature working. The **sqlFinance** editor presents a **user adjustable window** and the possibility to **change the size and style of the text**.

3. Menus

sqlFinance makes little use of **menus**. After the start of the application the menu is shown on the upper left screen. Only when the report form is shown the menu is changed to a choice of system options. The menu is mainly used to start **sqlFinance tools**. To make **application forms** visible, it is easier to use the **treeview** at the left of the screen.



sqlFinance main menu

Menu options on top of the screen:

Database

a. Connect
Establish a connection with a physical **local** or **server database**. Several connections may be opened **simultaneously** in a session. Use this **connection tool** or the **treeview** to toggle from one connection to the other.

b. Mandate
After connection with the physical database, select a **mandate**. The **list number** of the mandate determines which mandate appears **on top**. The mandate normally holds the accounting of **one single company** or the **consolidated accounts of a number of other mandates**.

c. Select
Main tool for selection in the target **server database**. On the top of the tool form the **current server connection**, **mandate** and **chart of accounts** are indicated.

Manage

a. Windows
A **list of currently open forms**. The list is presented in a **grid**. A **click** on a line will **activate the referenced form**. This tool has been replaced by the **treeview** on the left of the screen.

b. Records
The **records** tool is a **local register of form records**. The user can

remember with this tool **any records presented in an application form** and **recall** the record easily in later sessions. **Drag** a current form (with the right mouse key) over the records tool to **register en entry** with server/uniqnr and descriptive elements. Add a **comment** in the comments field, and modify the **title** if necessary. The entries in the list can be **grouped** according to **priorities** (1 to x). The **header** of the first record is used to **name the priority group**. The record list can be **exported**, transported on a data device and **entered in another sqlFinance application**. This is a valuable option for server applications, where users might access the database from **different installations** and use there **private records selection and notes**.

c. Notes

Notes is a simple tool for the handling of smaller **text elements** during a session. The size of the notes form is easily **adjustable**.

d. Options

A **list of parameters**. Some parameters can be **modified** by the user. The parameters are organised in a table. To a parameter always belongs a **name** and a **value**, whereas the value can be of **character** or **numeric** type.

- **ODBC_mgr**
  A variable to save the name of the **Microsoft ODBC Manager**, usually ODBCAD32X.EXE. The **Microsoft ODBC Manager** can be started from the operating system as well, writing "ODBCAD32X.EXE" into the command window. There are **several versions** of the **ODBC Manager** in the market. The use of some version can lead to very slow responses of the sqlServer. All shippings of **sqlFinance** include a copy of the preferred **ODBC Manager**, usually stored in the directory **"drivers".** ODBCAD32X.EXE is usually installed in the "windows\system32" directory.

- **Date-check**
  A system value **influencing the selection of data**. The value of the date-check can be **set in the selection tool** and is remembered for later sessions. **If the date-check is set to 1 ("on"), only records with on non-empty field "date_to" will be selected**. While **date_from** is usually set be default to the current date, **date_to** can be used to **check records** as **"done"** or **"at work"**. With a **double click** you can set the value of the **date_to** field to the **current date**. Confirm the update with the **"update"** key. In later selections, this record **will not appear any more**, if **date-check** is set to **"on"**. The **date_to** variable is mainly used in **bookings** and **contacts**.

- **Decimal**
  The number of **decimal digits** for fields of the type **money** (usually 2).

- **Autoboot**
  With autoboot set to **"on"** (1=default) the system will at **startup** run through **connection** procedures and show the **current mandate** and **all open forms left on the screen when the last session was closed**.

- **Mandate**
  The **name** of the **last used mandate**.

- **Translate Check**
  Parameter used to activate the **translation into a foreign language** (translation of **field names** and **messages**).

- **Textedit_size**
  Default value for the **character size** in the **sqlFinance Text Editor**.

View

a. Keylist
A Key Tool with a number of **keys**, corresponding to **menu positions**. Toggles the keylist visible or invisible. This tool is only kept available for compatibility and user's convenience.

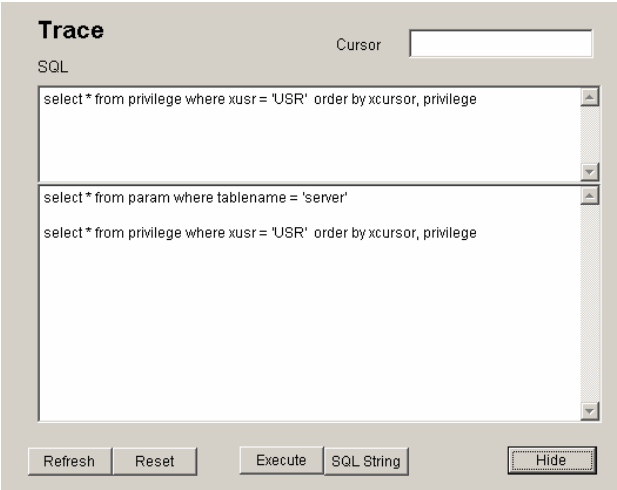
b. Treeview
Toggles the **treeview navigation tool** on the left of the screen **visible** or **invisible**. The treeview size can be adjusted with use of the mouse.

c. Clearing
A tool to **check links between booking records**. Dragging booking records over the clearing tool shows a **list of all linked bookings**. With help of this tool, **debit and credit transactions** can be worked through. In order to **link** booking records, drag the **"fields"** page section of one booking over the **"fields"** page section of another booking.

Tools

a. Engine
A powerful system tool managing **database access** and current variables setting. For debugging purposes only.

b. Trace
Tool to **analyse and test SQL statements**. The **last used SQL statement** is shown, as well as **all statements of the last transaction in a log**. The trace tool can be an efficient tool for **advanced users** familiar with **SQL Standard Query Language**. Use the **trace tool** to find out why some selections don't return any records.



the„Trace" tool

d. Replicator
Replication tool to **copy all add, update and delete operations from one computer to another**. In a **first step**, copy data from one computer and

the source data-base to the 'empty_DB', in a **second step** get the 'empty_DB' to the second computer (ex. with the help of a floppy disk or the internet), in a **third step** copy the 'empty_DB' data into the target database of the second computer.

e. Errors
Used for the **display of server error messages** (usually produced from **stored procedures**). All messages of the current session are traced in the error log.

f. Help
Manual pages, stored in the **info table** of the target database (set the **reference** field value to **'tips'**).

g. Info
Copyright and release information. The **copyright applies to this manual** as well.

sqlFinance is a **registered design** (**Design No. 126'732** at the **Swiss Federal Institute of Intellectual Property**). The software is **copyright protected**. Even though a part of the software is distributed for free, **you may not distribute sqlFinance without written consent of the owner or use it as part of a commercial product**. The Software is an **exclusive property of sqlFinance, Dr. Peter Halbherr**. Through the **download** you **don't become owner of the product**, but **you receive a license authorizing you to use the software for an unlimited period of time**.

h. Quit
Leave the application

.

4. Drag and Drop

To **transfer data** from one form to another, **drag and drop** operation is very efficient. In most cases, the forms must be **compatible** and have the **same set of fields**. **One record** or **several records** can be copied. The two forms may belong to **different mandates** or even **different databases** – eventually this way data may be copied from a **remote computer or server** to a **local computer or server**. To copy entire **sets of data** however, it is commended to use rather the **'replicator'** tool.

As an example, there might be **two equal forms connected to two different databases**, or **two booking forms of two different periods** (selected as details of two different batches). Dragging the fields frame over the target form will **automatically start the insertion of a new record and copy the field values**. The user then is **asked for confirmation** with a **click on the update key**.

In the **Customer Relationship management (CRM)** however, dragging **address, person or position** information over the **contac**t form adds each time the **respective data elements and a link**, leading to the full client contact information (as required by the "know your customer" specification).

In the following, find a list of **special drag and drop features**:

a. Accounts

To copy accounts from one chart to another, proceed as follows:

- With use of the selection tool, **open a form** with accounts of a chart.
- Again with the use of the selection tool, **open a chart form**, enter a new chart.
- On the **new chart form**, use the **detail key** to select **accounts** (no accounts will be found; the form is empty but **ready to accept new account entries**).
- **Drag page one of the first account form over page one of the new (empty) account form**.
- The software will **add a new record**, copy the account **field values**, and then **ask** if you just want to enter **one record** or **continue the entire list**: answer yes or no.

b. Products

Dragging product data from one form to another leads to a different behaviour. Products can be organised in a **Bill of Materials (BOM)**. The **Bill of Material** is a hierarchical organisation of **master products grouping a number of detail products**, whereas the detail product again may be master of other detail products. A **Bill of Material** can easily be viewed through **cascades of product forms**. Using the detail key, select connected detail products, and get back to master products with help of the master key.

In order **to set a new connection** between a master and a detail product, **drag** the **"fields" page** of one product form over the **"fields" page** of another product form. The software will enter a **new connection** (in the point table) and confirm with a message: **"connected"**. To **disconnect** a detail product, use the **"disconnect"** key of its form (the **product must have been selected as a detail of a master product** and not with use of the selection tool) to have this function **available**.

c. Processes

Process forms can be organized in **cascades** the same way like product forms. **Drag** the **"process"** over the **"contract"** to organize the contracts into **groups (pools)**, for example key accounts, small accounts, groups of special interest, etc.

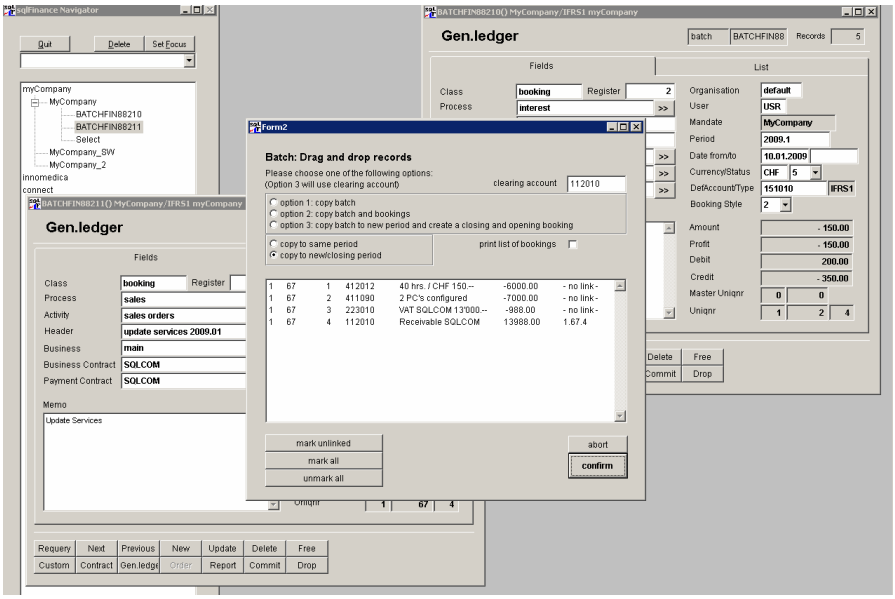
d. Batches

Very practical is the **copying** of a **batch** with use of **drag and drop**. Drag a batch from one form over another (batch) form. This will make appear a **dialogue form** offering the following **three options**:

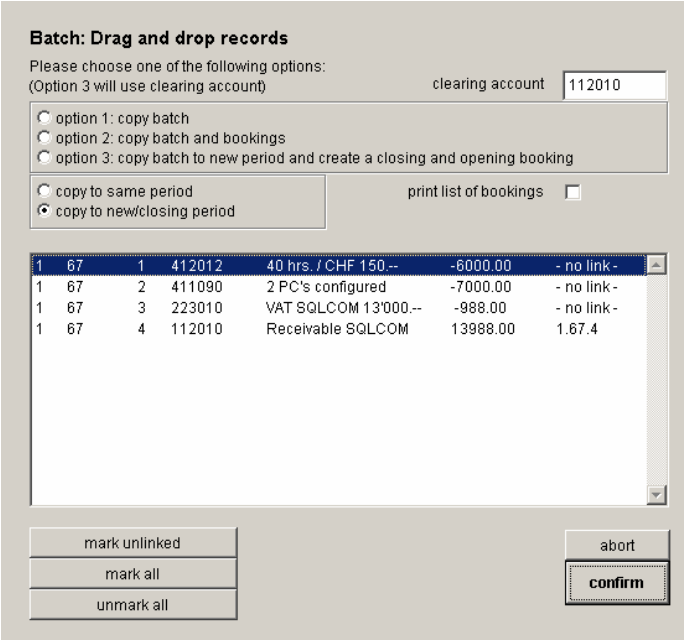
1. copy batch
2. copy batch and bookings
3. copy batch into a new period and create a closing and reopening booking

Option (3.) is mainly used for the **closing of open debit batches** at the **end of a yearly period**. This function requires the entry of a **transitory account** (e.g. a receivables account). Furthermore the user can decide if the closing and reopening bookings should include a detail list of the bookings in the closed batch.

For all three options the user can choose, if the batch should be copied to the **current period** or to the **new period**. The new period is usually the closing period of the current period (registered in the period form). In the transfer **all date entries are adjusted** (e.g. 30.6.2008 to 30.6.2009).



Overview of Drag and Drop dialogue window with batch forms



Detail view of the dialog form

e. Task Manager

Dragging any table over the **task manager** ("records" tool): Drag the "fields" page of any open form over the grid of the task manager to get a new entry right away. Use the **"SelectForm"** key to reselect the record.

f. Contacts

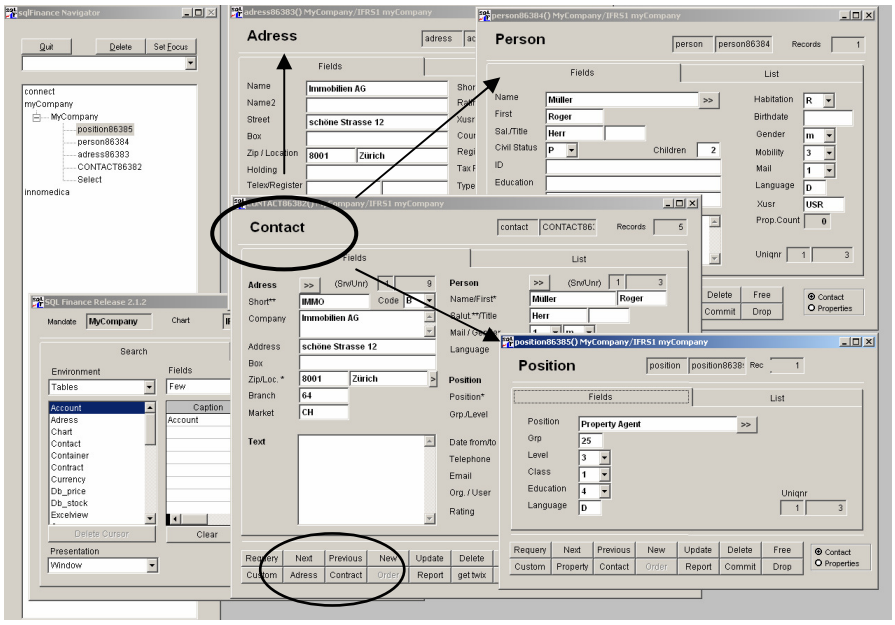
Drag **person, position and address** over the **contact** form: The **"contact"** table integrates information from the **"address", "person" and "position"** tables. Normally, not only name, position and address are copied to the "contact" table, but also the **reference numbers** (server/uniqnr) of these links.

The **contact** form can be opened as a **detail** of the **address** form. In the address, use the detail/contact key to get to the contact, and enter **"new"**. The record will **inherit many values** of the "address" entry. The contact entry will also store the system number of the address entry.

To link the **"person"** and **"position"** entry as well, drag the **"fields"** page of one of these forms over the **"fields"** page of the **"contact"** form: the values (as well as the reference number) will be copied. Note that the **"address"** entry may be dragged the same way, but it is more convenient to use an (address-contact) master-detail relationship.

g. Receivables

A **sales batch** is usually **cleared to zero** with a **receivable entry**, whose **amount equals the total of the bill**. After **reception of the payment**, drag the **receivable entry** over another **receivable form**. A **new entry** with the **negative amount** will be prepared. Just click **update** to commit. The two records, belonging to two different batches, are now **linked together**. In the **linked state** both records appear in **italic characters**. The link may be **erased** using the **"disconnect"** key. **Unlinked records** will appear in **normal characters** again. To get from one record to the (linked) other one, use the **detail key (or alt-F3)**. Use the **clearing tool** to get a **list of linked bookings**.



Contact form with reference to address, person and position

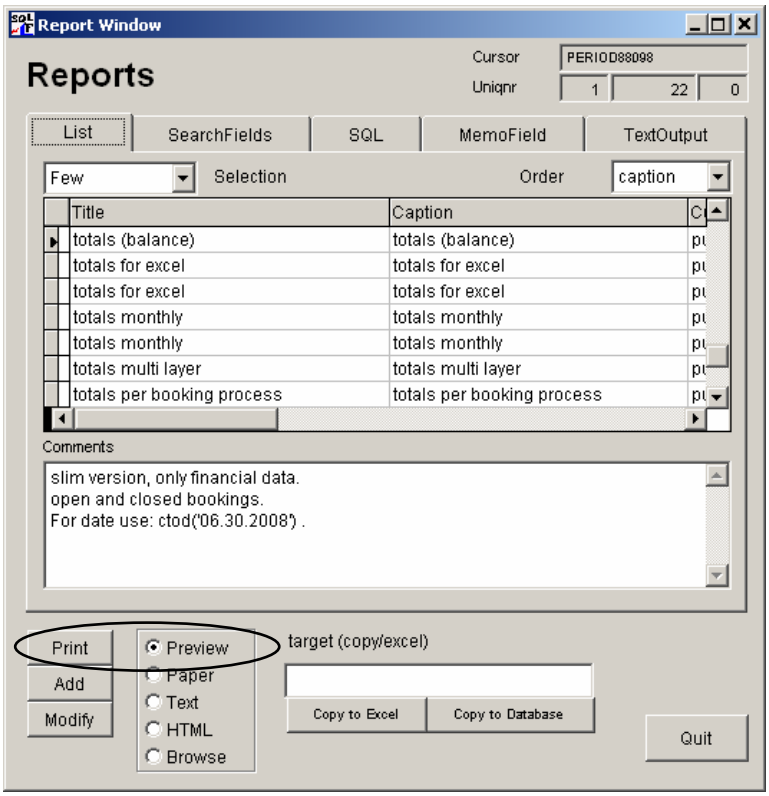
5. Reports

In order to start **reports**, consider the **data model**: the best reports are always found in the **master forms** and will select the **current master record** and the **details joined** to it. For example, a report based on a **period** entry will typically show all the **batches and bookings of that period**.

In **sqlFinance**, some reports can be started from the **selection tool**. In most cases however it is **much easier to start the report from a form**. In this case, the report can use information **starting from the current record** (joining further detail records) or information about the **list of current records**. The **list of current records** is used **if the report includes no SQL statement** (in the "fields" page of the report tool). If a **SQL statement** is provided, data is selected as a **detail to the current record**, as indicated in the **where clause of the SQL query**.

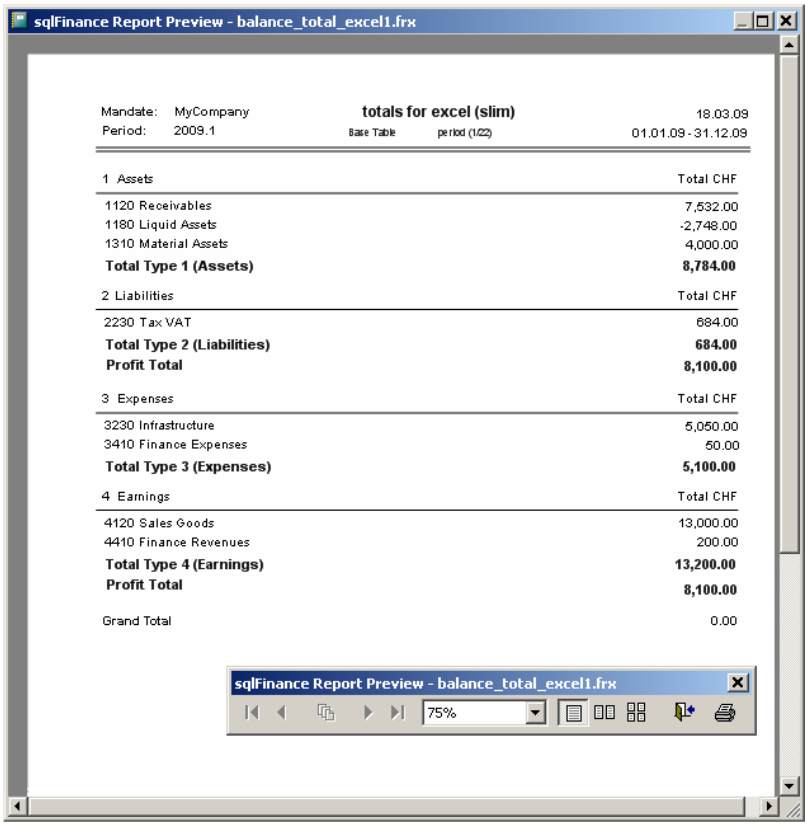
The **data list** is used, if in the report **no** further specifications for a **SQL statement** are present (see page "fields" of the report tool). As soon as an SQL statement is **given**, the search proceeds from the **current data record** and selects **connected details** according to the formulation of the query. Through **modification** of the SQL statement the user can add **additional fields** to the search or further limit the search in the **"Where-Clause".**

Without use of any additional **license** it is possible to **modify** the **reports** and their appearance. In order to receive for example a report of a period, one clicks in the period form on the button **"report"** and activates the appropriate **tool**.



The Reports tool started from the period form

In the report list, find the **report "balance totals"**. Click on the **"modify"** button in the "report" window to modify fields and appearance. **Confirm** with **ctrl-w**. Through **clicks** on the list view, the report may be directed to the preview, to the screen, to a printer, or in "ASCII" or "HTML" mode to a field on page three of the reports form. **Double click** on that field to edit or export the text to an external file. A more **condensed output** can be produced with the report **"totals for excel (slim)"**.



Sample report with balance totals

In addition to the report **"balance totals"** there is also the report **"balance total slim multi layer"**. This report can be used for **contribution accounting**. Accounts belong to **groups**, and the groups may be associated with different **partitions**. The contribution accounting will make apparent how the **margin** evolves as revenue and expense accounts belonging to another partition are added. As an example, **partition one** would show **gross margin** of sales minus cost of sales, **partition two** the **net margin** after deduction of additional overheads, and **partition three** the **company profit** after consideration of financials, taxes etc. The grouping of accounts can be used for the balance as well as for the profit and loss statement.

6. Keyboard Shortcuts

The following key combinations can be used in forms:

*CTRL-E:	edit text
*CTRL-S:	select
*CTRL-N:	next
*CTRL-P:	previous
*CTRL-A:	add
*CTRL-U:	update, save and quit in text form
*CTRL-D:	delete
*CTRL-Q:	requery
*CTRL-K:	keep
*CTRL-R:	report
*CTRL-G:	grid (page)
*CTRL-L:	list (page)
*CTRL-Z:	quit
*CTRL-F2:	master
*CTRL-F3:	detail
*CTRL-F1:	search (activate)
*CTRL-F10:	commit (in update mode: pay)
*ESC:	drop (and update the detail field with the value from the master table record)

V. Support & Contact

Need support for the installation or use of **sqlFinance**? Do not hesitate to contact us. Our team is ready to answer your questions and offer you an efficient solution for your business application.

Homepage: www.sqlfinance.com, www.ipag.ch
E-Mail: peter.halbherr@sqlfinance.com
Address: Dr. Peter Halbherr
Gesellschaftsstrasse 16
Postfach
3012 Bern
Phone : +41 31 311 04 27

Geschäftsstelle Zürich: Zollikerstrasse 144
Postfach
8034 Zürich
Phone: +41 44 383 88 22
Fax: +41 44 383 14 25

VI. Appendix

Example of a balance sheet with foreign currency and annual comparison (with permission of Innomedica Holding AG, Zug)

sqlFinance Report Preview - balance_total1.frx - Page 1

Mandate: Innomedica **totals (balance)** 19.03.09
Period: 2005.1 Base Table period (6/25) 01.01.05 - 31.12.05

1 Assets

Account		Total CHF
1150 Wertschriften (kurzfristig)		
115000	Wertschriften-Bestand Clearing	-10,000.00
115010	Aktien-Bestand Maerki	138,112.40
115011	Aktien-Bestand Ameritrade	488,637.59
		616,749.99
1150a eigene Aktien		
115020	eigene Aktien	386,387.28
		386,387.28
1180 Flüssige Mittel		
118070	Ameritrade	13,966.60
118080	UBS Konto CHF	6,058.01
		20,024.61
1180.1 Flüssige Mittel MB		
118060	CHFMaerkiBaumann	20,533.80
118061	USDMaerkiBaumann	9,838.16
118062	EURMaerkiBaumann	8,027.52
		38,399.48
1220 Immaterielle Vermögen		
122010	Immaterielle Vermögen	0.00
		0.00
1230 Sachanlagen		
123010	Sachanlagen	-0.00
		-0.00
Total Type 1 (Assets)		1,061,561.36

Page 1

sqlFinance Report Preview - balance_total1.frx - Page 2

Mandate: Innomedica **totals (balance)** 19.03.09
Period: 2005.1 01.01.05 - 31.12.05

2 Liabilities

Account		Total CHF
2110 Eigenkapital		
211010	Aktienkapital	800,000.00
211020	ges.Reserven	400,000.00
211030	Reserven f.eigene Aktien	445,734.28
211040	Freie Agio Reserven	110,317.22
211090	Gewinnvortrag	-870,001.02
		886,050.47
2120 Kreditoren		
222010	Kreditoren	15,877.25
223010	AHV	1,587.70
		17,464.95
2130 Fremdkapital (langfristig)		
213070	Wandelanleihen	73,704.00
		73,704.00
2190 Abgrenzung passiv		
219020	a.o.Trans.Passiven	5,380.00
		5,380.00
Total Type 2 (Liabilities)		982,599.42
Profit Total		78,961.94

Page 2

sqlFinance Report Preview - balance_total1.frx - Page 3

Mandate: Innomedica
Period: 2005.1

totals (balance)

19.03.09
01.01.05 - 31.12.05

3 Expenses

Account		Total CHF
3110 Einkauf		
311050	Rundungen	124.73
		124.73
3230 Infrastruktur		
323011	Infrastruktur	1,118.50
		1,118.50
3250 Abschreibungen		
325010	Abschreibungen	11,800.00
325020	Abschreibungen VST	2.05
		11,802.05
3260 Dienstleistungen		
326040	Vermögensverwaltung	15,838.25
326045	Fremdleistungen Normal Fix	5,380.00
326051	Internet	35.00
326054	Werbung	1,396.90
		22,650.15
3410 Finanzaufwand		
341020	Bankspesen	93.40
341040	Akt.Darlehen Zinsen Aufwand	439.50
		532.90
3510 Steuern		
351010	Steuern	234.50
		234.50
Total Type 3 (Expenses)		36,462.83

Page 3

sqlFinance Report Preview - balance_total1.frx - Page 4

Mandate: Innomedica **totals (balance)** 19.03.09
Period: 2005.1 01.01.05 - 31.12.05

4 Earnings

Account		Total CHF
4210 Währungsgewinne		
421010	Währungsgewinne	16,143.06
		16,143.06
4410 Finanzertrag		
441010	Finanzertrag Zinsen	5.93
		5.93
4420 Handelsertrag		
442020	Ertrag Wertschriftenhandel	1,300.00
442021	Bewertungsgewinn eigene Aktien	92,915.10
		94,215.10
4421 Handel eigene Aktien		
442022	Handel eigene Aktien	5,060.68
		5,060.68
Total Type 4 (Earnings)		115,424.77
Profit Total		78,961.94
Grand Total		0.00

Page 4